Háskólinn á Akureyri
University of Akureyri

Viðskipta- og raunvísindadeild
Faculty of Business and Science

Setup of a DNS Server with Dynamic Updates

Final Year Project
2009

Eric Erlandsson
Frans Englund
Jenny Schulze (Háskólinn á Akureyri)
Mattias Sjöblom

UNIVERSITY OF GOTHENBURG

Setup of a DNS Server with Dynamic Updates

Bachelor of Science Thesis

FRANS ENGLUND ERIC ERLANDSSON
JENNY SCHULZE MATTIAS SJÖBLOM

University of Gothenburg
Department of Computer Science and Engineering

Göteborg, Sweden 2009
Report No. 2009:26

CHALMERS

Setup of a DNS Server with Dynamic Updates

Bachelor of Science Thesis

FRANS ENGLUND
ERIC ERLANDSSON
JENNY SCHULZE
MATTIAS SJÖBLOM

Department of Computer Science and Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2009
Report No. 2009:26

Setup of a DNS Server with Dynamic Updates

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden

Abstract

Today the need for dynamic reconfiguration of IP addresses increases. This is mostly due to the increased mobility among Internet connected devices. The purpose of the project was to develop a solution for dynamic DNS with updates initiated by the client. There are already various commercial solutions available today, but they have some limitations. The goal was to implement a system based on already available components with only minor adaptations. This report describes an implementation based on a RADIUS server (FreeRADIUS), which is accessed through a web interface (daloRADIUS). The authentication process is secured by a server certificate. Due to time constraints the implemented result should be considered as an early prototype. The conclusion of the project is that the concept works and that it is possible with some effort to solve the problem in a relatively simple way.

Sammanfattning

I dag ökar behovet av att dynamiskt kunna konfigurera IP-addresser. Detta beror mestadels på en ökad mobilitet bland Internet-anslutna enheter. Syftet med projektet var att utveckla en lösning för dynamiska DNS uppdateringar som initieras av klienten. Det finns redan i dag ett antal kommersiella lösningar på detta problem, men de har vissa begränsningar. Målet var att implementera ett system baserat på redan tillgängliga komponenter med bara några få justeringar. Den här rapporten beskriver en implementation baserad på en RADIUS server (FreeRADIUS), som nås via ett web-gränssnitt (daloRADIUS). Autentiseringsprocessen skyddas av ett certifikat. På grund av tidsbegränsningar bör det implementerade resultatet ses som en tidig prototyp. Slutsatsen av projektet är att konceptet fungerar och att det går att lösa problemet med viss ansträngning, på ett relativt enkelt sätt.

Table of Contents

1. Introduction 1
1.1 Background 1
1.2 Purpose 2
1.3 Limitations 2
1.4 Problem Description 2
1.5 Method 3

2. Technical Background 5
2.1 DNS 5
2.2 RADIUS 7
2.3 LAMP 7
2.4 Certificates 7
2.5 Iptables 9

3. Implementation 10
3.1 Equipment 10
3.2 Network Topology 11
3.3 Design 11
3.4 Service Configuration 12
3.4.1 Basic Implementation 13
3.4.2 RADIUS 14
3.4.3 The Script 15
3.4.4 Apache 15
3.4.5 Certificates 16
3.4.6 Firewall 16

4. Results and Discussion 17
4.1 RADIUS 17
4.2 DDNS 17
4.3 Certificates 18
4.4 Test and Verification 18

5. Conclusions 19
5.1 Advantages and Disadvantages of the Solution 19
5.2 Future Work 20
5.3 Final Conclusion 21

References 22

Appendix A - Contribution Report 25

Glossary

BIND - Berkeley Internet Name Domain, a DNS Server.
DDNS - Dynamic Domain Name System, DNS with dynamic updates whenever a client changes its IP address.
DHCP - Dynamic Host Configuration Protocol, a service that supplies network configuration information for hosts.
DNS - Domain Name System, a service that translates names to numerical addresses and vice versa.
DNS lookup – Name query for an address of a domain name to a DNS server, the DNS server performs the work and answers.
FQDN - Fully Qualified Domain Name. The complete name of a computer including the domain. "computer.chalmers.se" is a FQDN while "computer" is not a FQDN.
FreeRADIUS - Open source implementation of a RADIUS server.
IP - Internet Protocol, the network protocol used to communicate between units in the Internet.
IP address - The numerical address to a unit in the network.
ISC - Internet Systems Consortium, Inc., maintains BIND and DHCP implementations.
MAC address - Media Access Control address. An identity on a physical network interface. The identity should be unique and used in link-layer switching.
NTP - Network Time Protocol. Protocol for time synchronization over IP.
NTPD - NTP daemon. Unix implementation of a NTP client/server.
RADIUS - Remote Authentication Dial In User Service. A network protocol used for centralized Authentication, Authorization and Accounting (also known as the AAA concept). It is used when managing computers or people that have to access some kind of network service.
Router - A computer or a standalone networking device that connects subnets with each other, and routes traffic between subnets.
SQL - Structured Query Language. A computer language used in programming and managing relational databases.
SSH - Secure Shell, a secure protocol used for accessing a computer remotely. It is widely used in the Linux/Unix world, but there are implementations available for other operating systems as well.
SSL/TLS - Secure Sockets Layer/Transport Layer Security are cryptographic protocols implemented on top of TCP/IP. They are used to provide an encrypted point to point connection over an insecure network (e. g. The Internet).
Switch - A layer-2 network device that extends the network physically but does not route traffic between different subnets it does only work within a subnet.
TSIG - Transaction SIGnature, symmetric key system, mostly used in DNSSEC.
TTL - Time To Live. A time value telling how long some kind of data is valid.
Watchdog - A function that supervises some other function and reacts on changes.

1. Introduction

Dynamic Domain Name System (DDNS) is a method to change Domain Name System (DNS) entries in a DNS server automatically, without any need for manual reconfiguration. To fully gasp the content of this report basic network knowledge is required. It is also needed to understand the purpose of the product. More technical details will be explained in chapter 2.

1.1 Background

Nowadays many client computers often change their Internet Protocol (IP) addresses. Owing to the shortage of IP addresses many users have dynamic IP addresses instead of a static one. Also, a lot of people own a laptop or other portable device connected to the Internet. This furthermore increases the need for mobility since they will often join different networks and will obtain a different IP address each time.

A problem arises if you want to host a service like a File Transfer Protocol (FTP) server or a web server. When users connect to a service they want to use the same information every time they connect. The way to accomplish this is either to have a static IP address or a static Fully Qualified Domain Name (FQDN) e.g. "mywebserver.ddns.com". Since a static IP address is not always possible the solution has to be a static FQDN. Usually the FQDN is mapped to a static IP address in the DNS server, but for a laptop this normally does not work.

The ideal solution is called DDNS (Albitz & Liu, 2001). DDNS permits automatic changes of DNS entries in the name server. In that way a new IP address can be mapped to the FQDN, allowing the server administrator to make use of dynamic IP addresses. In this way the user who wants to connect to a server only needs the FQDN.

DDNS would also allow clients to keep their hostnames even if the IP configuration is done dynamically. Especially when they are moved in smaller environments, e.g. a computer lab in a university. In this case the DDNS would allow an administrator to move computers without any reconfiguration at all.

Similar solutions
DynDNS is a DNS service provided by Dynamic Network Services Inc. (DynDNS.com, 1997). Its primary concept is to allow dynamic updates of DNS entries. This is done by client software that periodically checks whether the IP address of the host has changed. If a change is detected, the changed IP address is sent to the DynDNS server and the DNS entry is updated. In this way the computer hosting the service can always be accessed although its IP address changes. A similar implementation is provided by no-ip.org (No-IP, 2000).

1.2 Purpose

The main purpose of the project is to implement a DDNS solution that works across the Internet. The only requirements should be that the client is connected to Internet and has an IP address configured. The initial sub goal is to implement this DDNS solution in a lab at the Lindholmen campus. If a computer with a registered name is moved to a new network consequently its IP number in the DNS database should automatically be reconfigured, allowing it to keep the same FQDN. This implementation is supposed to use only standard software packages, have client authentication and be easy to implement.

The goal of the project is to find a complete and working solution to the problem. The solution might be usable outside the scope of the project, but mainly to satisfy the necessary functionality for the specific computer lab.

1.3 Limitations

Only already available software will be used, which means that there will be no developing of any new software. Small adaptations might be done to the available software, but they will be kept to a minimum.

The solution will only be tested on:

- Debian servers
- Debian and Windows XP Pro clients
- a wired network with approximately 20 computers

There is a limited availability of hardware, consequently testing will only be conducted on available hardware. In a commercial point of view a larger test base would be required. Doing large scale testing is outside the scope of the project, since there are limited resources available.

1.4 Problem Description

The task is to find a working solution for the problem defined and do a basic implementation. The implementation does not need to be complete and may leave room for further improvements. The project focuses on finding and implementing a solution that is as simple and as basic as possible.

Initially the hardware that is available needs to be configured. This consists of building a test network and setting up the routers and switches. When there is a test network available the test computers need to be configured. This work consists of connecting the computers to the test network and then installing a suitable operating system. For the servers this would be Debian Linux. The clients would be a mix of Windows XP and Debian Linux. Some of the computers might already be installed with a suitable operating system.

The next thing to do would be to get the basic services running. In this project the key service is the DNS system. A Dynamic Host Configuration Protocol (DHCP) server is also needed for testing purposes. Both the DNS and DHCP service should be made redundant (i. e. a primary/secondary pair should be used for both services). The initial purpose is to get basic DNS and DHCP running to make testing and experimenting with various solutions possible.

The next phase would be to start looking for a suitable solution. This will include finding a suitable authentication solution and adding auxiliary services. It is also time for decisions of what should be included in the project implementation and what to put in the list of possible improvements.

Finding a solution is one of the major challenges of the project. There are some commercial solutions that the project can use for inspiration. However, most of them rely on some kind of host agent which this project would like to avoid, if possible. There is also a challenge in minimizing the transition time whenever a client changes its IP number since DNS relies heavily on different caching techniques.

The solution itself should provide some way of allowing a client to authenticate itself. Whenever a client authenticates the corresponding DNS entry should be updated. The simplest form of authentication would be a login and a password. The project will also look into the possibility of using some kind of certificate based authentication.

The solution should be able to handle that the client could be anywhere on the Internet. As soon as the client has acquired an IP address it should be able to update its home DNS server with this new IP address. At that moment the DNS server should update the records with the new IP address. This will allow the client to roam the entire Internet as long as it can acquire an IP address and connect to its home DNS server.

Finally the project management itself is also a challenge since the project has limited resources. At various points, the project will have to make decisions on what should be included in this project and what should be left to possible future project.

1.5 Method

The methodology of the project consisted of three basic parts. First of all the problem was specified, relevant sources identified and a preliminary design developed. This included looking for solutions and evaluating them. In the second part the project used a prototyping approach to gain continuous feedback from the customer or in this case the supervisor. The second part used several iteration steps as shown in Table 1.

Table 1: Method

iteration	**achievement**
1	get running DHCP and DNS service
2	setup RADIUS
3	setup website to work with RADIUS
4	secure server with firewall
5	implement certificates on website
6	perform final testing

During the iteration process, the supervisor was consulted and some testing was made. These steps lead to some changes during the project. The implementation in the project mainly consisted of editing configuration files. As they were stationary on the servers, there was no synchronization problem and no version tracking was needed. Backups were made of the relevant files on a regular basis. The final part was to compose everything and finalize the product.

As the project goals changed the initial design had to be revised. This was done more in an implementation on demand than in a coordinated way. As a result much time was spent in following dead ends. As an example much time was paid to find a Remote Authentication Dial-In User Services (RADIUS) client for Local Area Networks (LANs) supporting certificates if the website could use certificates.

The final stage was simply a matter of rounding up the project and finalizing everything. This included deciding what to include in the current implementation and what to set as future possibilities. It also included finalizing the documentation of the project.

Since the project had four members most of the work was divided into smaller parts. Most parts of the project were done either by one member of the project individually or by two members working together. It was discovered to be a much more efficient way, allowing all active members to contribute in a more efficient way. This allowed the other member to do other work, allowing the project to be more time efficient.

2. Technical Background

This section gives a brief description of software and protocols that are used in the solution. How they were used in this project is described in chapter 3.

2.1 DNS

The DNS is used to translate host names and service names (e. g. www.chalmers.se) to numeric IP addresses (e.g. 129.16.221.8), which is a more suitable format for computers. In short this is done by a lookup in the DNS server's database, and if not found the server will contact other DNS servers to get the correct IP address for the requested lookup (Figure 1).

IP addresses to other DNS servers and hosts will be cached in the local DNS server performing a lookup. How long an address is valid in the cache is decided by the TTL value of the IP address. TTL is set when the address is added in its authoritative DNS server's database. This cache function results in that commonly used addresses and domains are often found in the cache. With cached information the time of the lookup decreases and less data traffic is produced.

In a full DNS lookup without any cached information, the local DNS server will request a root name server for an IP address to the correct top level domain DNS server (e.g. .se, .com). The top level server, which knows what lower level DNS servers' IP addresses are, redirects the local DNS server further down the hierarchy of DNS servers. This proceeds between the local DNS server and other DNS servers until the IP of the requested hostname is known or results in an error. The local DNS server also sends the correct address or an error to the client that requested the DNS lookup.

The DNS server contacted last in a lookup which is responsible for a portion of the name space delegated to its organisation, which is called the DNS zone. This authoritative DNS server has the address saved in its database along with the TTL value.

An example of a DNS lookup where the top level domain server is known by the local DNS server is illustrated in Figure 1.

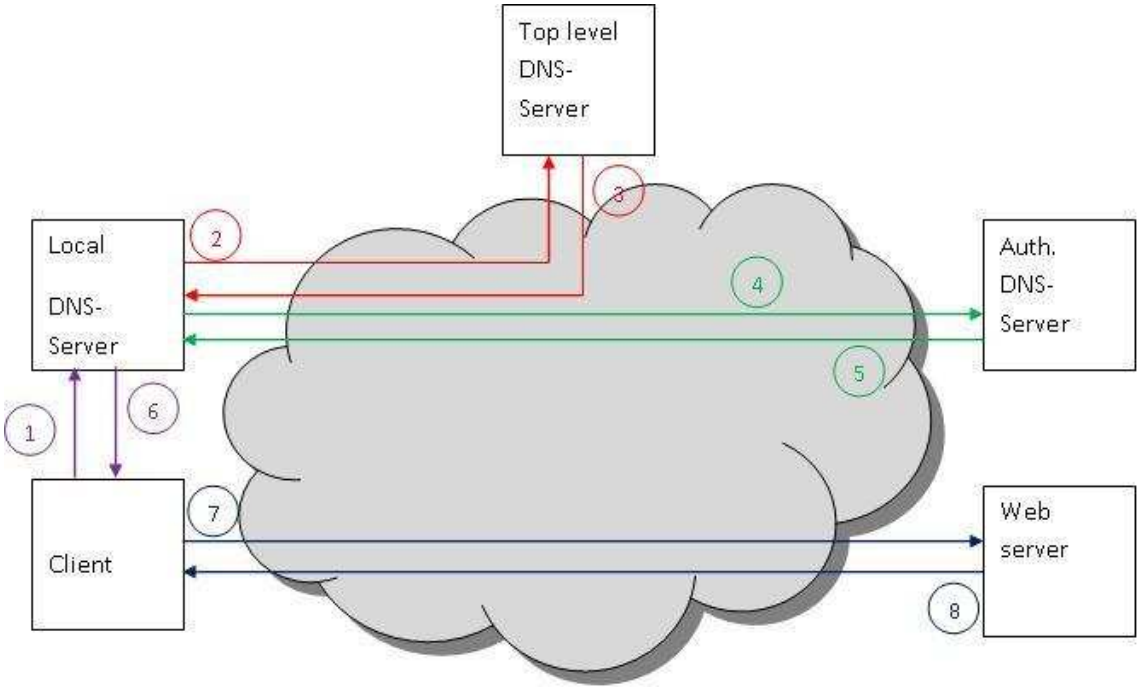

Figure 1: Standard DNS lookup

1. Client asks for an IP address to a certain name of for example a web server.
2. Local DNS server asks a top level domain server for an address to a lower level DNS server.
3. The top level domain server answers with an IP-address to an authoritative DNS server.
4. Local DNS server asks the new DNS server for the address of the web server.
5. The server was an authoritative DNS server thus it answers with the correct IP address.
6. The IP address is forwarded to the client.
7. Now the client can ask for the web page from the web server since it got the exact address.
8. Data exchange between client and web server starts.

DNS Security Extensions (DNSSEC)

DNSSEC is a software solution that adds security to the DNS protocol (Arends, 2005). It provides origin authentication, data integrity and authenticated denial of existence for the DNS protocol. In regular DNS lookups the client can ask for digital signed DNS data sent from all DNS servers used in the lookup. A server of one level higher position in the hierarchy can verify the origin of the DNS server below. Signing can be done with a symmetric key (e.g. TSIG) or an asymmetric key (e.g. SIG) (Albitz & Liu, 2001).

DNSSEC is also used when updating the DNS database which is useful over an open network (Albitz & Liu, 2001). If redundancy is needed, i.e. the DNS implementation is based on two or more computers; in that case it is vital to use DNSSEC for sending the DNS database between the servers to maintain full security. In this case it is not only encryption that is needed, other security solutions are useful e.g. Message Authentication Code (MAC).

2.2 RADIUS

The RADIUS is an Authentication, Authorization and Accounting (AAA) network authentication protocol (Rigney, C. et al, 2000), (Cisco, 2006). RADIUS is used for authentication and can easily be combined with other software. The client sends a request to the Network Access Server (NAS) in order to use a specific resource. The NAS forwards the request to the RADIUS server, which replies with a challenge for the request (e.g. password). If the client's password is found in the RADIUS server's database and several other credentials (e.g. IP address) are met access is granted, else access is denied. The RADIUS protocol is in widely spread usage which means there are a large number of client implementations. This allows most platforms to make use of the solution.

FreeRADIUS is a server implementation of the RADIUS protocol. It comes with a PHP web interface which can be modified at users' discretion.

daloRADIUS provides a web interface for administration of FreeRADIUS. It is written in PHP and JavaScript, thus it can easily support various database formats (daloRADIUS, 2008). Also it provides a login interface for users.

2.3 LAMP

LAMP is a solution stack of software (ONLamp.com, 2001). That means it is a package of software which together works as a full solution. LAMP is an acronym for Linux, Apache, MySQL and PHP (also Perl or Python). These components together works as a web server. The LAMP concept is used in this project.

Apache HTTP Server is open source software which is used for web page hosting (Laurie, B & Laurie, P, 1999). It is the most common web hosting application used (Netcraft, 2009). There are many modules for apache which extends the possibilities of usage.

MySQL is an open source implementation of a relational database (MySQL, 1998). The distribution also contains the tools necessary for editing and managing databases. As the name implies it uses the database computer language SQL.

PHP, which is the chosen scripting language in this project, is a server scripting language designed for dynamic web pages i.e. the content of the web site is changing.

2.4 Certificates

Certificates are used for secure connection between a client and a server. When the client tries to connect to a web page the server will send a certificate containing information that the client can use for encrypting further traffic exchanges with the server. If the client accepts the certificates the server grants access to the requested web page and all data traffic goes under encryption.

TLS/SSL handshake with certificates
If a client wants to connect to a secured web site, it first sends a "client hello" message to the server (Figure 2). This message includes information about the TLS/SSL protocol version, what ciphers and compression method the client supports, and some initial random numbers. The server send the same information about itself to the client in response, it also adds its certificate ("server hello"). The server might request a client certificate to verify the client, but this is optional.

In any case client calculates a pre-master secret out of the two random numbers and sends it to the server ("client_key_exchange"). The pre-master secret is encrypted with the public key of the server. Now both the server and the client can calculate a master secret out of the pre-master secret, by applying a combination of the MD5 and SHA algorithms on the two random numbers and the pre-master secret. This master secret is used to calculate six further keys, three for each side. One key for writing, one key for signing and the last one as initialisation vector for block encryption. All these new keys are symmetric, to limit the time needed for decryption. Finally a control message is sent containing all messages send previously but encrypted. The server returns the same information. From now on every further message is encrypted.

This function is included in Apache as a module called mod_ssl (Engelschall, 2001).
If the signing process is done by the server itself the resulting certificate is called a self-signed certificate and the user is informed about this by its web browser. On the other hand if it is signed by a registered certificate provider no extra information is provided.

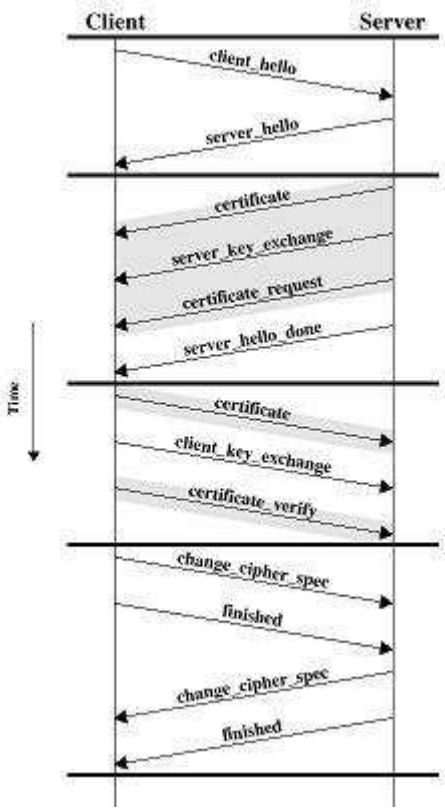

Figure 2: TLS/SSL handshake (Panko, 2003)

2.5 Iptables

Iptables is open source software, often preinstalled with many linux distributions (netfilter.org, 2002). Iptables is used to decide what should be done with incoming or outgoing packets, they can e. g. be dropped, accepted or logged. Decisions can be made based on several cases. For example which interface or what port that is used, whether the connection was already established or not. It is also possible to filter by protocol. Iptables is often used as firewall software but can also be extended to other purposes e.g. routing and forwarding of data traffic.

3. Implementation

This section describes details about the designed solution, the available equipment and the configuration of services. The major focus in the project was to use readily available components and compose them into a single working system. The project mostly used packages available from Internet, as this simplified the installation. All software used was under free license and was open source.

3.1 Equipment

For testing purposes the project had access to two Cisco 2600 series routers and a DLink switch for the network setup. The two routers were preconfigured by our supervisor. Only DHCP forwarding was added to the configuration (Cisco Documentation, 2000). That means that the broadcast packages sent by a client when asking for a IP address will be forwarded to the subnet where the DHCP server is. The two routers were connected through a serial line to supply the test network with a total of four router interfaces and consequently four attached subnets. Only three of them were actually used, one subnet for each server subnet and one subnet for the test clients.

There were a number of computers available for development and testing. These computers were Intel based (x86) standard desktop computers. Two of them were used as servers and the remaining ones were used as test clients. It was decided to run the servers with Debian Linux as operating system.

3.2 Network Topology

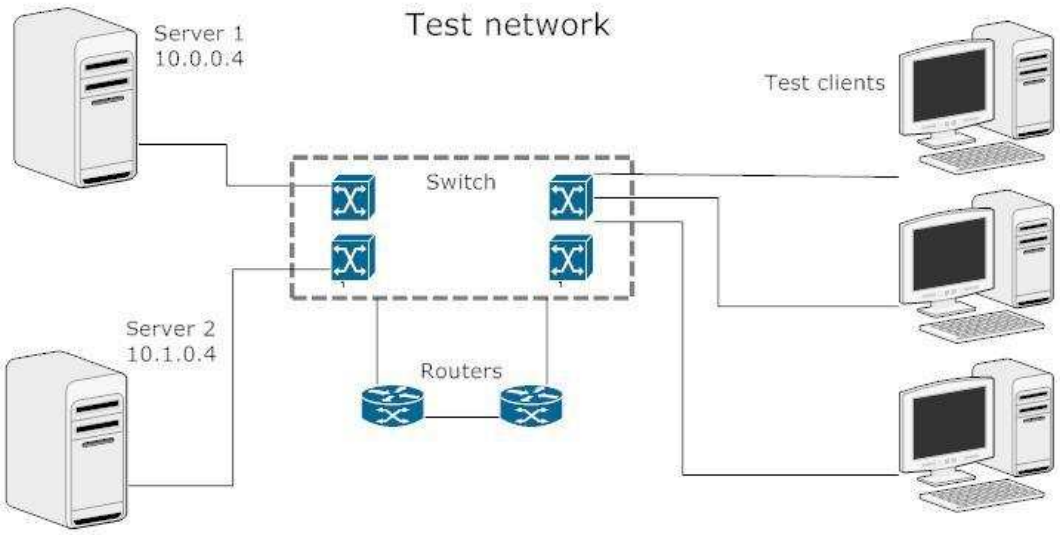

Figure 3: Topology of test network

The basic network topology is quite simple (Figure 3). The solution itself does not rely on the network topology therefore it is used for testing purposes only. The servers got fixed IP addresses, 10.0.0.4 for master and 10.1.0.4 for slave as they remain in place. Except for the servers and router interfaces, all addresses in the different subnets are provided by a pair of DHCP servers.

It is preferred to have the primary and the secondary server of a service on different networks. This gives some protection against local network hardware failure. However, when being far from any service (in a networking point of view) this has a minor impact since there will be a multitude of network equipment that might fail between the computer and the name service.

3.3 Design

The technical design is kept as simple as possible (Figure 4). All necessary client information (user name, password, host name, domain name, last IP) is stored in a MySQL database. The client is able to update its IP in the database through a web page which asks for an account name and a password. The password transmission is protected by SSL/TLS (https). The account data is evaluated through a RADIUS server (FreeRADIUS) that verifies the login data against the stored one in the database. If the username and password provided are correct, the current IP of the user is updated in the MySQL database and a script is evoked. The script is provided with host name, domain name and IP address of the user that just logged in. Then the script communicate with the DNS service, deletes the old entry for the combination of host name, domain name and adds a new entry with the new IP address.

The reason a RADIUS server was used is that it is easy to add other ways of authentication at a later time. For example, RADIUS has built in support for authentication through cryptographic certificates and there are RADIUS clients available for most platforms.

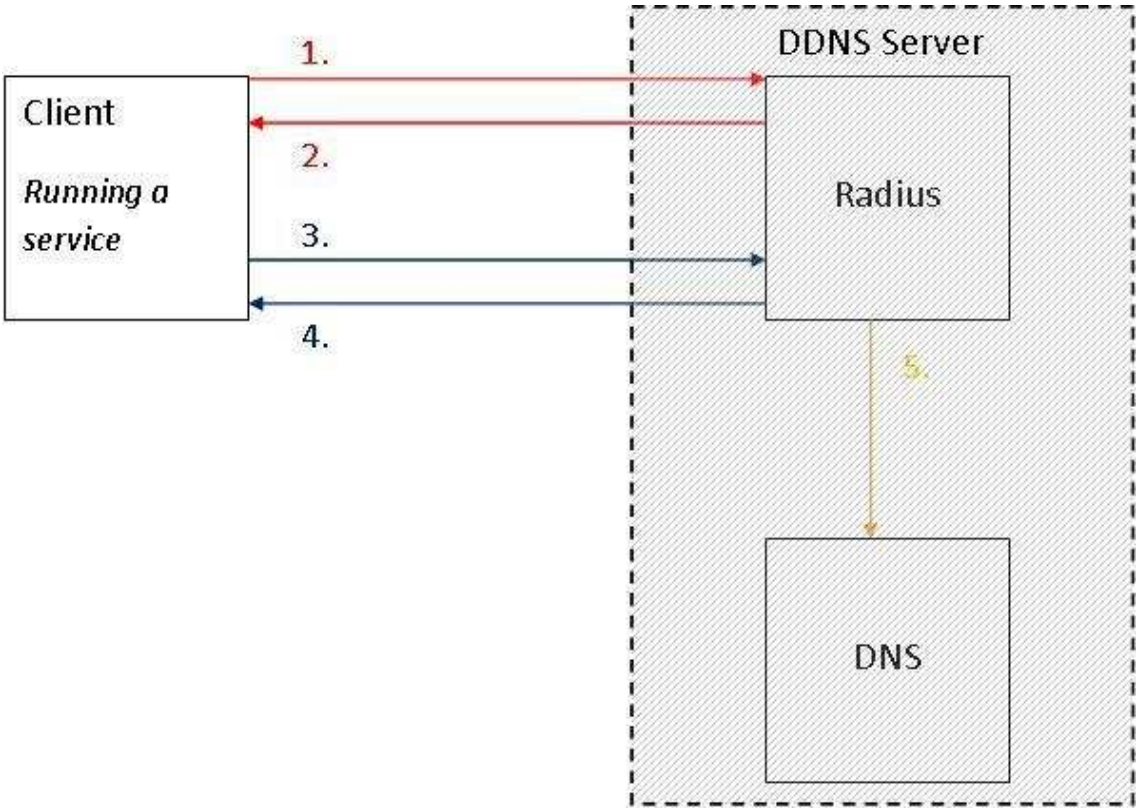

Figure 4: Implementation of the solution

1. Client asks for connection
2. RADIUS and client set up a secure connection
3. Client provides user name and password
4. Authentication OK
5. Update of clients new IP in the DNS database

3.4 Service Configuration

The implementation was started with basic services like DHCP and DNS. Later on FreeRADIUS with MySQL, daloRADIUS and Apache where set up. Lastly, a firewall was added.

3.4.1 Basic Implementation

The DNS and DHCP servers were implemented at a basic level. The project chose ISC BIND for the DNS part of the project and ISC DHCP for DHCP (Internet Systems Consortium, 1997). Both of them are widely used implementations of DNS and DHCP. Furthermore both BIND and DHCP have already made packages for Debian (Debian Manual, 2006). Both were configured manually to work as they should in our small network.

DHCP Servers
Initially the DHCP servers were intended to be part of the solution. The idea was to use the DHCP server as initiator of the updates. Since this would require control over the DHCP server, the mobility would be limited to networks under the direct control of the solution. As the design goal was to allow mobility over the entire Internet, this solution was abandoned about halfway through the project. However, the DHCP servers were kept for testing purposes throughout the project.

The DHCP servers were configured as a master and a slave server. If the slave has not gotten any signal from the master for more than 30 seconds, it assumes the master has failed and the slave takes over (Heinlein, 2005), (jny.dk, 2007). This setup ensures that if only one server fails the clients can still get a valid IP address in a network known to the routers. The master and slave splits the addresses of the different subnets between them to avoid address collisions. If one server goes down, the other server takes over the responsibilities for the other server's addresses. This operation is reversed when the faulty server comes online again.

DNS
Also the DNS servers were implemented redundant, i. e. there is a master and a slave server (Albitz & Liu, 2001). The default configuration only contains standard data. There was need to create a new zone that we gave the name warning. In order to allow for updates the zone property "allow-update" was set accordingly. The two servers synchronize each other periodically and on changes in the DNS table (semicomplete, 2008), i. e. when the master gets updated through the script it immediately sends the new entry to its peer. For this synchronization a TSIG key is used to increase security.

NTP
Both DNS and DHCP require the time difference on the different servers to be within a certain limit, especially the TSIG includes a time stamp which has to be accurate. In order to ensure DHCP and DNS to work correctly it is necessary to synchronize the time on the servers, therefore they were set up as peers using NTPD version 4.2.2 (ntp.org, 2000). The local clock on Server2 acts as reference clock. Since ordinary PCs are used as test computers the Local Clock in this case is the bios clock, which is the built-in clock in the hardware of the computer. This allows Server1 and Server2 to have the same time even though the real time will be inaccurate. As the test network is completely isolated from the Internet and there is no access to a better time source, this is sufficient. The accuracy of the bios clock is very poor, but the goal is to keep a global clock in synchronization and not to have accurate time.

3.4.2 RADIUS

Since it is aimed to have a perfect match between security and simplicity of implementation, it was decided to make use of the RADIUS protocol (Rigney, C. et al, 2000). It provides security features like encrypted passwords and can make use of certificates. Furthermore its accounting abilities can be used to limit access to the service and track the use of the service. This gives enough security for most small to medium sized networks.

FreeRADIUS

It was decided to use FreeRADIUS as RADIUS server (FreeRADIUS, 2006). First an older Debian package (version 1.3) was installed, but it did not support certain password features, e. g. plain text passwords that are good for testing. Consequently it was updated to version 2.1, which created some inconsistency in the system, as example some language warnings turned up when installing new packages. Still there were some features regarding certificates missing. The source code was consulted and needed files were copied.

MySQL was used to store user data (e. g. passwords, user names). FreeRADIUS is also able to interact with Oracle and a variety of other databases, but they were not that easily available as MySQL (Elmasri & Navathe, 2007). FreeRADIUS provided a script which made it easy to add all necessary tables to the database.

daloRADIUS

daloRADIUS is a web interface for FreeRADIUS. It requires some additional tables in the database in order to manage user accounts, like login/logout time or for more commercial use like billing information. These were easily added using a provided script.

It also provides an interface to administrate the FreeRADIUS database tables. This means that there is no need to interact with MySQL directly (i. e. writing SQL statements), just to fill out a form particularly made for RADIUS.

In order to be useful for IP updating information the userinfo table was modified. It normally holds contact information like name, address, company. As such information is not needed for this project, the existing rows were reused. The rows containing first name, last name and email address now hold host name, domain name and IP address respectively. The table header has not been changed, only the file that maps the table header to HTML output. This technique was used because there are multiple references to the table header and it would have been a tremendous work to modify every single file.

daloRADIUS does not seem to read all information stated in the FreeRADIUS configuration file. This information includes the "Exec-Program-Wait" entry, that can be used to execute a script before or after the authentication process. Therefore in order to execute a script that could update the DNS records it was needed to update the dologin.php file of daloRADIUS. This file is used to authenticate the user. After the user is authenticated, its IP address is obtained, and the data containing host name and domain name is extracted from the userinfo table, mentioned before. This information is send to the script (HowtoForge, 2008). The file dologin.php now becomes responsible also for updating the IP address in the MySQL database.

3.4.3 The Script

The script is responsible for updating Server1. It uses nsupdate and a TSIG key. It only updates the actual zone. First the DNS entry is cleared then the new DNS entry is added. The information needed is provided by the PHP script in daloRADIUS.

```
#!/bin/bash
USER="$1"
IP="$2"
HOSTNAME="$3"
DOMAIN="$4"
# update DNS entry
nsupdate  << EOF
server 10.0.0.4
key updatekey <some-secret-key>
zone warning
update delete $HOSTNAME.$DOMAIN
update add $HOSTNAME.$DOMAIN. 600 IN A $IP
send
quit
```

Figure 5: The script file “DNS-update”

3.4.4 Apache

The user needs to log in independently from the computer (s)he uses at the moment. Having a specific client program may create some inconvenience for the user, why a more widely interface like a web site is needed.

Nowadays Apache is the most widely available web server for Linux. First an appropriate Debian package was used, but when it came to certificates, an additional package was needed. This resulted in some incompatibilities and a problem, the PHP code was not processed correctly anymore. Therefore Apache and PHP where installed from source code, resulting in having the most up to date version of Apache - 2.2.11.

3.4.5 Certificates

The login interface uses a certificate. Since there is no Internet access available for the servers, it is impossible to sign them by any trusted agency. If the client decides to accept the certificate the normal key negotiation process takes place.

3.4.6 Firewall

In order to enhance the level of security for the servers a firewall was implemented on each server. It was chosen to use iptables, as it came already installed with the system. The configuration was kept simple, all incoming ports that were used are allowed and all others are blocked (LinuxHelp, 2002). There is no filter on outgoing ports. Allowed incoming ports include:

Table 2: Firewall

Port	**Service**
80	http (website)
443	SSL (certificate)
1812 - 1817	RADIUS
67	DHCP
53	DNS
22	SSH (administration)

4. Results and Discussion

The project used a number of services in the implemented solution. Some of them are critical to the solution while others are of a more auxiliary nature. Critical services include daloRADIUS, FreeRADIUS, Apache, MySQL and the DNS server on Server1. All services on Server2 are auxiliary, as it only implements a slave DNS server, DHCP and NTP. DHCP is not at all needed in the solution, as clients could get an IP address by manual configuration as well. As NTP is only used to synchronize Server1 and Server2 it is also auxiliary.

The core services create a critical chain of dependency while the auxiliary services provide helper functions. The helper functions are less critical and the solution will in most cases survive without the helper functions for a short period of time.

4.1 RADIUS

Radius uses two services for its help, an Apache server and a MySQL database. These services were implemented on Server1 and they are working as they should do. However we had plans on making the services fully redundant. To achieve full redundancy the Apache server, the MySQL database as well as the RADIUS server should have been installed on both servers. Due to time constrains we chose not to make these services redundant.

The RADIUS implementation consists of two parts, the FreeRADIUS server and the daloRADIUS web interface. The daloRADIUS interface contains too many functions for our needs. We only needed a login box and a page to view hostname, domain name and IP address. The extra information in the daloRADIUS interface does not affect our implementation, especially not on a version used in a test environment. If it would be implemented in a real world solution it would be better to make use of this information. Moreover, additional settings and information might also be built in.

When logging in through the daloRADIUS interface the FreeRADIUS database is read for validation of the user. If the login is approved a script is run which updates the DNS server and the database with the client's new IP address. This is working exactly as it was intended to do.

4.2 DDNS

The standard functions of the DNS servers are working just as they should and they are redundant. If the master DNS server (Server1) goes down the slave server will take over. However since the RADIUS server, Apache server and the database were only implemented on Server1, the DDNS solution is not redundant. If Server1 goes down it will be no longer possible to update the records. Server2 will still work as a standard DNS server without the functionality to update the records.

4.3 Certificates

As mentioned in subsection 3.4.5 the server certificate was not signed by a trusted agency. However if the client chooses to trust the server the normal key negotiation process takes place. We also planned on using client certificates but were unable to find a solution for it. The only solutions we found were for WLAN.

4.4 Test and Verification

Originally the plan was to test the solution in one of the lab rooms on Lindholmen with approximately 20 computers but due to lack of time we only tested the solution in a simpler network with three clients. The results would probably not differ but a success on a larger test would have shown even more that the solution does work.

5. Conclusions

The DynDNS solution is similar to our solution. The main difference is that we chose to use RADIUS as the outward interface. This allows our solution to work with certificates as well in a future version. There are RADIUS agents available for most common operating systems, which eliminates the need of a special agent.

5.1 Advantages and Disadvantages of the Solution

The solution is easy to implement as it uses a combination of well documented standard services like Apache, MySQL and RADIUS. Also, the very basic services like DNS and DHCP are redundant, enabling the user to obtain address resolution and IP changes even if one of the servers fail.

Another good aspect is the use of RADIUS. Since there are standard clients available in many operating systems, a special agent will not be necessary. There is still need for an update script, in this case embedded in FreeRADIUS and not in daloRADIUS as today. Such a script is relatively trivial to develop, but on the other hand would require a lot of testing, since a lot of agents have to be included.

Parts of the project that could be improved are to implement redundancy for the web, RADIUS and MySQL servers. Since they rely one on each other which means that there is a chain of dependency, if one of them fails, it will be impossible for the user to update its DNS entry. Furthermore, all of these services are located on the same physical computer, which makes that computer a single point of failure. If this solution is to be deployed in a large network, it would be necessary to increase the redundancy by having more than two copies of each service. However, for testing purposes, our implementation was enough.

A serious weakness of this implementation is the amount of outdated packages, due to an old CD distribution of Debian. Sometimes, there was a need for up to date packages. Mostly, this was needed for special options in Radius and certificates in Apache. In case of installing the packages which created conflicts reinstallation was forced. This also resulted in some minor warnings.

The project also implemented a solution based on server certificates. As the certificate is self-signed the user gets a warning message about this nature. On the other hand certificates greatly enhance security as the communication is encrypted.

The solution only supports IP version 4. IP version 6 is becoming more popular and it is already in wide spread use in some parts of the world. A future solution would need to support both IP version 4 and IP version 6.

5.2 Future Work

There are several functions we have thought of during the project that would add more functionality to our solution. However, due to time constrains this was limited to the functions we have implemented.

Certificates
One major function that would be useful is to be able to use client certificates, and certificates within RADIUS (Panko, 2003). Now only the client is able to verify the identity of the server. However, if using client certificates, also the server could identify the client. Certificates are more secure than a simple password and user name combination which are more prone to shoulder surfing and eavesdropping. Certificates could be implemented both in the web server and in the RADIUS server. Certificates also simplify automated updates through an agent or a script.

Agent
It is also a good idea to have an automated agent installed on the client. The agent would automatically detect an IP address change of the client and immediately send a request to update the address record in the DNS server through the RADIUS server. This is manageable both through the web interface, i. e. the client connects to the website and makes use of the HTTP protocol or directly through RADIUS. The RADIUS method is probably more efficient as it does not need a helper application like the web interface. Security will not be harmed as both RADIUS and Apache can use certificates. Moreover, there are RADIUS clients available to most common operating systems that are significantly reducing the administration work.

Redundancy
If the web server would fail, no client can connect to it and therefore it is not able to update its DNS entry in case its IP address changes. Resulting in no client service is reachable, as the IP address stored in DNS entries is outdated. The same problem arises if the RADIUS server would fail. This problem can be solved by adding a redundant solution for both the web service and the RADIUS service. Depending on the implementation of the redundancy there might be a need for some round robin DNS entries and a watchdog updating the entries if one service is down.

Another problem arises if the database fails, the user cannot be identified although the web interface is reachable and RADIUS is giving correct replies but probably a "user cannot be identified" error message. To enable redundancy for the database, one solution would be to have two database instances that are synchronized somehow. The synchronization could be implemented as a master/slave system or by having an external master for example.

IP Version 6
Since IP version 6 is already in commercial use in several parts of the world, adding support for IP version 6 to the solution will be necessary for future use. The solution will have to be able to support both IP version 4 and IP version 6 at the same time, since a client might move between IP version 4 and IP version 6 networks. This work would require more research to be done and it would also be necessary to do more testing since IP version 6 software is less mature. Part of the challenge would be to ensure proper behaviour with a client using both IP version 4 and IP version 6.

5.3 Final Conclusion

There were some ups and downs during the project and we had to do a major redesign once. We also made several smaller adjustments to the design during the project. The product itself is working as intended. We feel confident that the solution would work perfectly in the lab rooms at Lindholmen, which we could not test due to lack of time. It would need further improvement to be really applicable in a commercial point of view. Hopefully, the results of this project will be utilized in the future, maybe by us or some other group.

References

Albitz, P & Liu, C., 2001. DNS and BIND. 4th Ed. Sebastopol, California: O'Reilly Media, Inc.

Arends R. et al., 2005. DNS Security Introduction and Requirements. [Online] The Internet Society
Available at: http://www.ietf.org/rfc/rfc4033.txt [Accessed 18 May 2009].

Cisco, 2004, How Does RADIUS Work? [Online] (Updated 18 May 2009)
Available at:
http://www.cisco.com/en/US/tech/tk59/technologies_tech_note09186a00800945cc.shtml [Accessed 18 May 2009].

Cisco Documentation, 2000. Routing IP. [Online] (Updated 20 Dec 2003)
Available at:
http://www.cisco.com/univercd/cc/td/doc/product/software/ssr83/rpc_r/48383.htm
[Accessed 18 May 2009].

daloRADIUS, 2008. About. [Online] (Updated 19 May 2009)
Available at: http://www.daloRADIUS.com/?q=node/1 [Accessed 19 May 2009]

Debian Manual, 2006. How to setup DHCP. [Online] (Updated 18 May 2009)
Available at: http://www.crazysquirrel.com/computing/debian/servers/dhcp.jspx
[Accessed 18 May 2009].

DynDNS.com, 1997. DynDNS.com: DNS Hosting, E-mail Delivery, VPS Hosting and Other Services. [Online] (Updated 18 May 2009)
Available at: http://www.dyndns.com [Accessed 18 May 2009].

Elmasri, R & Navathe, SB, 2007. Fundamentals of Database Systems. 5th Ed. Boston, Addison-Wesley.

Engelschall, RS, 2001. User Manual [Online] (Updated 7 April 2009)
Available at: http://www.modssl.org/docs/2.8/ [Accessed 18 May 2009].

FreeRADIUS, 2006, HOWTO - FreeRADIUS Wiki [Online] (Updated 7 January 2009)
Available at: http://wiki.freeradius.org/HOWTO [Accessed 18 May 2009].

Heinlein, P., 2005. Failover with ISC DHCP. [Online] (Updated 28 Mars 2009)
Available at: http://www.madboa.com/geek/dhcp-failover/ [Accessed 18 May 2009].

HowtoForge, 2008, Authentication, Authorization & Accounting with FreeRadius & MySQL backend & web based Management with daloRADIUS [Online] (Updated 18 May 2008)
Available at: http://www.howtoforge.com/authentication-authorization-and-accounting-with-freeradius-and-mysql-backend-and-webbased-management-with-daloRADIUS [Accessed 18 May 2009].

Internet Systems Consortium, 1997. Internet Systems Consortium. [Online] (Updated 18 May 2009)
Available at: https://www.isc.org/ [Accessed 18 May 2009].

jny.dk, 2007. ISC-dhcpd failover article. [Online] (Updated 18 May 2009)
Available at: http://www.jny.dk/?q=node/10 [Accessed 18 May 2009].

Laurie, B. & Laurie, P., 1999. Apache: The Definitive Guide. 2nd Ed. Sebastopol, California: O'Reilly Media, Inc.

LinuxHelp, 2002. Firewall Script. [Online] (Updated 18 May 2009)
Available at: http://www.linuxhelp.net/guides/iptables/ [Accessed 18 May 2009].

MySQL, 1998. MySQL :: The world's most popular open source database. [Online] (Updated 18 May 2009)
Available at: http://www.mysql.com/ [Accessed 18 May 2009].

Netcraft, 2009. March 2009 Web Server Survey. [Online] (Updated 29 May 2009)
Available at:
http://news.netcraft.com/archives/2009/03/15/march_2009_web_server_survey.html
[Accessed 29 May 2009].

netfilter.org, 2002. netfilter/iptables project homepage - The netfilter.org project. [Online] (Updated 6 Apr 2009)
Available at: http://www.netfilter.org/ [Accessed 18 May 2009]

No-IP, 2000. No-IP - Dynamic DNS, Static DNS for Your Dynamic IP. [Online] (Updated 18 May 2009)
Available at: http://www.no-ip.com/ [Accessed 18 May 2009].

ntp.org, 2000. Reference clocks. [Online] (Updated 18 May 2009)
Available at: http://www.ntp.org/ntpfaq [Accessed 18 May 2009].

ONLamp.com, 2001, What is LAMP? [Online] (Updated 18 May 2009)
Available at: http://www.onlamp.com/ [Accessed 18 May 2009].

Panko, RR, 2003. Corporate Computer and Network security. Upper Saddle River, New Jersey: Prentice Hall.

Rigney, C. et al., 2000. RFC2865 - Remote Authentication Dial In User Service (RADIUS). [Online] The Internet Society
Available at: http://www.ietf.org/rfc/rfc2865.txt [Accessed 18 May 2009].

semicomplete, 2006. Dynamic DNS and DHCP - Easy to do, and you'll thank yourself later. [Online] (Updated 18 May 2009)
Available at: http://www.semicomplete.com/articles/dynamic-dns-with-dhcp/ [Accessed 18 May 2009].

Appendix A - Contribution Report

Responsibilities
At the start of the project, we split the project into areas of responsibility. Eric was responsible for structuring of the reports. Frans took on the role as leader. Jenny and Mattias were responsible for the technical development of the solution.

Early part of the project
Eric and Frans did most of the planning and structure work on the planning report. All of us contributed to the writing of the report. We also shared the work of looking for articles, books and web sites that we could use in the project.

Implementation
In the early part of the project all of us contributed equally to the technical implementation. Jenny and Mattias were responsible for the structure and later in the project Jenny and Mattias did most of the work with the technical implementation while Eric and Frans still made minor contributions.

Final Report
Eric was responsible for the structure of the report and did most of the formatting work. Frans wrote a lot of text to the report during the entire project. In the last part of the project we all contributed to the report and we also split up the report in to areas of responsibilities.

Eric: 4, References
Frans: 1, 5, Contribution Report
Jenny: 3, 4
Mattias: 2, 3, Figures

Presentations
Half time presentation: Frans and Mattias
Posters: Eric and Mattias
Final oral presentation: Eric and Jenny
Final oral opposition: Frans and Mattias

Summary
It was an advantage to spread the different types of work among different persons to increase the efficiency. This also allowed us to do several things in parallel and while we might have wished to add more functionality to the solution, it kept the workload on a reasonable level during the whole project. All in all, we feel that we managed to share the work equally and that we all contributed to project on a similar level.