TAMPERE UNIVERSITY OF TECHNOLOGY

SAKU KÄSSI

IPV6 HOME NETWORK - CONNECTING HOME NETWORK NAME SERVICE TO THE INTERNET DNS

Master of Science Thesis

Examiner: Professor Jarmo Harju
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering
on 5 June 2013.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Degree Programme in Information Technology
KÄSSI, SAKU: IPv6 home network - Connecting home network name service to the Internet DNS
Master of Science Thesis, 77 pages, 10 Appendix pages
December 2013
Major subject: Communication Networks and Protocols
Examiner: Professor Jarmo Harju
Keywords: IPv6, IPv4, home network, name service, DNS, automatic configuration

Current home networks are very simple containing only a few devices. As the number of devices connected to the home network increases, there is no reasonable way for a user to access devices using only IP addresses. Due to the exponential growth of devices connected to the Internet, the addresses of the current IP version are however soon to be depleted. A new IP version has already been implemented in the Internet, containing a very large amount of addresses compared to the current IP version. Addresses in the new IP address version are also much longer and more complicated. Therefore it is not reasonable to try to use IP addresses alone to access devices anymore.

The previous facts force to implement a name service to the home network. Name service is quite similar to that used in the Internet, although the home network version should be much more automatic and user friendly. This means that users do not have to type IP addresses anymore to be able to access services, but they can use meaningful names like in the Internet. The first objective of the thesis is to examine methods to implement as automated name service as possible to the home network.

Second objective is to examine connecting the home network name service to the Internet name service. Accomplishing this allows users to access services at home from the Internet. This has to be made in a secure manner to protect the integrity and authenticity of the user information.

A live experiment of the thesis concentrates to the second objective of the thesis by establishing the connection and transferring the name service information between home network and the Internet name service.

The study and the live experiments indicate that there is still work to be done before the two objectives can be fully accomplished. At the moment there is no convenient way to automatically name devices at home. Connecting to the Internet name service involves also quite a lot of effort, thus requiring more than basic computing skills from the user.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
KÄSSI, SAKU: IPv6-kotiverkon liittäminen Internetin nimipalveluun
Diplomityö, 77 sivua, 10 liitesivua
Joulukuu 2013
Pääaine: Tietoliikenneverkot ja protokollat
Tarkastaja: professori Jarmo Harju
Avainsanat: IPv6, IPv4, kotiverkko, nimipalvelu, DNS, automaattinen konfiguraatio, tietoturva

Nykyiset kotiverkot ovat hyvin yksinkertaisia sisältäen vain muutaman päätelaitteen. Jo nyt yhä useammat kodin laitteet pystyvät liittymään verkkoon, ja laitteiden määrä kasvaa koko ajan. Näin ollen kohta ei ole enää mielekästä käyttää kotiverkon palveluja pelkän IP-osoitteen avulla.

Toisen haasteen tuo nykyisin käytössä olevien IP-osoitteiden loppuminen. Uusi IP-versio on jo käytössä ja tarjoaa nykyään samat palvelut kuin nykyinen versio sekä huomattavasti suuremman osoiteavaruuden. Osoiteavaruuden kasvun myötä myös osoitteen pituus kasvaa ja se monimutkaistuu.

Edellä mainitut asiat, eli laitemäärän kasvu ja tästä johtuva IP-osoitteiden muuttuminen aiheuttavat sen, että kodin laitteita ei pystytä enää hallitsemaan pelkillä IP-osoitteilla. Näin ollen on järkevää ottaa käyttöön Internetissä käytössä olevan nimipalvelun kaltainen palvelu myös kotiverkkoon. Tämä nimipalvelu ja palvelujen automaattinen mainostus muille kotiverkon laitteille on diplomityön ensimmäinen tavoite.

Toinen tavoite on tarkastella, miten kotiverkon nimipalvelu pystytään liittämään Internetin nimipalveluun. Tällä mahdollistetaan kotiverkossa olevien palvelujen käyttö myös kotiverkon ulkopuolelta. Yhteys kotiverkon nimipalvelusta Internetin nimipalveluun pitää pystyä suojaamaan siten, että kotiverkon tietojen muuttumattomuus pystytään takaamaan. Tästä osuudesta tehtiin myös diplomityöhön kuuluvat laboratorio-osuus.

Sekä teoria että laboratorio-osuus osoittivat, että tällä hetkellä ei ole vielä lähellekään valmista ratkaisua, jolla edellä kuvatut ominaisuudet pystytään toteuttamaan riittävän käyttäjäystävällisesti. Standardointityö aiheen osalta on vielä kesken, joten laite- ja ohjelmistovalmistajat eivät pysty vielä toteuttamaan keskenään yhteensopivia toimintatapoja.

PREFACE

Initially, I was given two choices for the topic of the thesis the easier and the harder one. Although I got to choose the topic ultimately, I am glad that my supervisor from TeliaSonera Finland, Mr. Jyrki Soini suggested me to take the topic that we both saw to be the harder one. I would like to express my gratitude to Mr. Soini for supervising me and helping me to challenge myself to explore the topic that was not familiar to me previously at all.

For the implementation part, I would like to thank Mr. Mika Paananen and Mr. Timo Snellman for providing me tips and configuration aid, especially during the lab. I also thank Mrs. Riina Annala for helping me during the writing process.

My dear family, I owe you a lot. You have supported and helped me on your own remarkable way. In recent months the spare time we have had has been quite limited, but now the thesis is done and I do not have to share it anymore.

Tampere, November 13, 2013

Saku Kässi
Ruopionkatu 4 F 45
33800 Tampere

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

ALQDN	Ambiguous Local Qualified Domain Name
AP	Access Point
ARP	Address Resolution Protocol
AXFR	Authoritative Transfer
GB	Giga byte
BIND	Berkeley Internet Name Domain
CA	Certificate Authority
CER	Customer Edge Router
CIDR	Classless Inter Domain Routing
CPE	Customer Premises Equipment
DAD	Duplicate Address Detection
DHCP	Dynamic Host Control Protocol
DHCPv6	Dynamic Host Control Protocol version 6
DLV	DNSSEC Lookaside Validation
DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
DNS-SD	Domain Name System Service Discovery
DoS	Denial of Service
DS	Delegation Signer (Resource record)
DNSSEC	DNS Security extension
FQDN	Fully Qualified Domain Name
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICANN	Internet Corporation for Assigned Names and Numbers
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IKEv2	Internet Key Exchange version 2
IPX	Internetwork Packet Exchange
IPsec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IXFR	Incremental Zone Transfer
ISC	Internet Systems Consortium
ISP	Internet Service Provider
KSK	Key Signing Key
LLMNR	Link-Local Multicast Name Resolution
MAC	Media Access Control
mDNS	multicast Domain Name System

NAT	Network Address Translation
NETBIOS	Network Basic Input/Output System
NS	Neighbor Solicitation
NSEC	Next SECure
NSEC3	Next SECure3
NTP	Network Time Protocol
ORO	Option Request Option
POOF	Passive Observation Of Failures
PKI	Public Key Infrastructure
PTR	Pointer (Resource record)
RADV	Router Advertisement
RAM	Random Access Memory
RFC	Request for Comments
RR	Resource Record
RRset	Resource Record set
SD	Service discovery
SEP	Secure Entry Point
SOA	Start of Authority
SLP	Service Location Protocol
SSDP	Simple Service Discovery Protocol
SRV	Service (Resource record)
TTL	Time to Live
UDP	User Datagram Protocol
UI	User Interface
ULA	Unique Link Local
ULQDN	Unique Locally Qualified Domain Name
UPnP	Universal Plug and Play
TB	Tera byte
TCP	Transmission Control Protocol
TKEY	Transaction Key
TSIG	Transaction SIGnature
TXT	Text (Resource record)
WINS	Windows Internet Name Service
WLAN	Wireless Local Area Network
ZSK	Zone Signing Key

1 INTRODUCTION

IPv4 world has come to an end. More and more devices are connecting to the Internet. Use of the Internet is spreading fast especially in developing countries. At the same time the people that have been using the Internet longer connect more devices to it. Ten years ago a single computer formed the home network, but now there may be multiple devices capable of connecting to the Internet per each family member. At home, networking is already enabled in numerous household appliances and building automation systems and more will be enabled in the near future. All these facts lead to the situation that there will be no more IPv4 addresses available. Simultaneously, IPv6 networks have been built and the same services that are available in IPv4 become available also in IPv6. Soon there will be no reason why a home user could not migrate home network to IPv6.

In IPv4 network services and devices can be managed using IP addresses only, since an IPv4 address is much simpler and shorter than an IPv6 address. Using IPv6 at home and connecting more and more devices to the home network soon comes the point, where one cannot manage all services and devices reasonably by using IPv6 only. Something has to be done. The situation can remedied using a concept that everybody uses daily, when browsing the Internet, the name service.

Sometimes services and devices at home should be accessed from the Internet. So name service used at home must be connected to the Internet name service or DNS.

The objectives of the thesis were to examine possibilities to build as automatic home network as possible and examine how to establish secure connection between the name service at home and the Internet DNS. Automation should be understood as means for devices to obtain the needed configuration by themselves in order to communicate with other devices and the Internet. Devices should also be able to announce services, they can provide, by themselves.

The theory part of the thesis consists of the following chapters. In Chapter 2 the evolution of automatic network configuration is discussed, starting from proprietary protocols and moving to IP compatible implementations. Chapter 3 describes the current and the future network architectures. Chapters 4 and 5 focus on name service protocols that could be used to enable name service at home and connect it to the Internet DNS.

The implementation part consists of three chapters. Chapter 6 introduces methods that have been developed to enable automatic naming and methods to connect home network name service to the Internet DNS. Chapter 7 discusses best practices for home network mainly from the home user point of view. Chapter 8 introduces the lab home network that was implemented according to the findings of the preceding chapters.

Results and conclusions of the thesis are represented and discussed in Chapters 9 and 10.

2 EVOLUTION OF AUTOMATIC NETWORK CONFIGURATION

Since computers became networked there have been many different types of network protocols and topologies. Nowadays the IP is the de facto network layer protocol, regardless of the network equipment and operating system manufacturer. Before IP became the de facto, networks were more operating system dependent. Almost every operating system manufacturer had its own network protocol. Some of these proprietary protocols had automatic configuration mechanisms. Users just connected a computer to the network and it started to communicate with the other device using the same protocol. Since the IP has become the protocol that every manufacturer supports, it can be used to build networks using hardware and software from different manufacturers. The IP also has created challenges to the automatic network configuration. The lack of automation has been compensated by inventing some semi-automatic mechanisms, such as DHCP (Dynamic Host Control Protocol). In this chapter we discuss the history of automatic configuration of networks and how it has been done in the IPv4 world.

2.1 Automatic configuration networking

In an ordinary IP network, an administrator has to configure IP addresses manually to hosts in advance or set up a DHCP server to distribute IP addresses to hosts. Name service has to be set up as well for hosts to be able to use mnemonic names instead of IP addresses to reach the services. Sometimes these actions are not possible or sensible. Zero configuration networking or Zeroconf is a set of techniques to create an operational IP network without the need of manual configuration. This means that no servers are needed for the DHCP or DNS (Domain Name System) services. Computers, printers etc. connected to the network can use its services instantly without any assistance from an administrator. (IETF 2008)

IETF formed the Zeroconf working group in 1999. It was addressed to develop specifications for:

- Interface Configuration (IP address, network prefix, gateway router)
- Name-to-Address Translation
- Service Discovery
- Automatic allocation of Multicast Addresses
- Sufficient security features to prevent networks from being any less secure than networks which do not use Zeroconf protocols. (Cheshire 2012; IETF 2008)

Working group did not come to a consensus in any of the above requirements. Yet, in 2004 it published a protocol specification titled Dynamic Configuration of IPv4 Link-Local Addresses. IETF has published it as RFC3927. (Cheshire 2012; IETF 2008)

Even before 2004 Apple, Microsoft and Novell had their own protocols for Zeroconf like easy-to-use networks. In 1980's Apple developed a protocol called AppleTalk. User only needed to connect computers together with a LocalTalk cabling and the network was up and running. Later in 1990's the same was accomplished with the AppleTalk-over-Ethernet protocol. The latest, wireless, version is the AirPort, which is based on IEEE 802.11 standard. Microsoft and Novell had their own versions for Windows computers as well, Microsoft NETBIOS and Novell IPX. (Cheshire 2012)

Everyone had own solution, but cross-platform communication, for example between Apple and Microsoft computers was impossible. The IP was and is the only cross-platform communication protocol that all of the operating systems support. Techniques used in AppleTalk are used to develop an easy-to-use protocol that uses IP as network layer protocol.

Since 2004 there has been some development. The first three items of the IETF requirements for Zeroconf working group have been solved with the following techniques, although the titles have changed.

- "Interface Configuration…" was replaced by "Allocate addresses without a DHCP server", which is defined in the RFC 3927 - Dynamic Configuration of IPv4 Link-Local Addresses
- "Name-to-Address Translation" was replaced by "Translate between names and IP addresses without a DNS server", which is defined in the RFC 6762 - Multicast DNS
- "Service Discovery" was replaced by "Find services, like printers, without a directory server", which is defined in the RFC 6763 - DNS-Based Service Discovery. (Cheshire 2012)

The last two on the list, Automatic allocation of Multicast Addresses and Sufficient security features are yet to be solved. The overall requirement, gathering all requirements listed, is that they must be able to coexist with the configured network and must not interfere it anyway. (Cheshire 2012)

2.1.1 Automatic address allocation

When connecting to a Zeroconf network, a host has to determine its IP address by itself. There is no DHCP server or network administrator to configure IP settings to the host. IP addresses used in Zeroconf are defined in RFC 5735. The RFC allocates an IPv4 address range of 169.254.0.0/16 for the link local use (Cotton & Vegoda 2010, p. 4). IANA has reserved the first and the last 256 addresses from the 169.254.0.0/16 range for future purposes, so the actual link local address range is 169.254.1.0 -

169.254.254.255. (Cheshire et al 2005, p. 10) The link local address range for IPv6 is fe80::/64 (Hinden & Deering 2006, p. 11).

The rest of the paragraphs of this section are based on the RFC 3927 written by Cheshire et al. (2005, pp. 10-13). A host uses the pseudo-random number generation algorithm when choosing its address. The algorithm is used to prevent two hosts from choosing the same address. Recommendation is to use host’s IEEE 802 MAC (Media Access Control) address as seed value for the algorithm, so a host most likely selects the same link local address every time it is booted and two different hosts unlikely select the same address. This is very practical for debugging and other operational reasons as well.

After selecting the address, the host must verify that the address is unused in the network segment it is connected to. The host performs the test by probing the network segment with ARP (Address Resolution Protocol) message. ARP can only be used in networks that are based on the Ethernet technology. The RFC describes the probing test for other link-layer technologies only by stating that similar claim and defend mechanism should be used.

After the ARP probe is sent out, a host has to wait for defined period of time for ARP reply messages. A host may also receive a request message that has the selected address in the sender IP address field. If such a message is received, host must repeat the address selection and probing process. Probing process must also be done when after sending out an ARP probe host receives a similar ARP probe but sender hardware address is not its own. The situation occurs when two or more hosts attempt to claim the same IPv4 link local address at the same time. The address selection and probe test process must be repeated until no conflicting messages are received. If a host has not received any conflicting ARP reply or probe messages by the defined period of time, it determines that has successfully claimed the selected IPv4 link local address.

The last thing to do is to announce the claimed IP address, so that other hosts in the network segment can add or update the claimed IP address with the correct hardware address to their ARP cache. A host sends out an ARP announcement message stating the address it has claimed.

2.1.2 Name service and automatic host name selection

In the local network the Internet DNS might not be available always, for instance due to outage on the ISP connection. When this happens users in local network cannot use the Internet name service, since DNS servers are usually located in ISP’s network.

Zeroconf provides a solution to the problem above. Zeroconf enables name service in a local network without the need of an administrator involvement or connection to an ISP. It uses .local. pseudo-top-level domain in a local network to distinguish between the local and other domain names. Conventional DNS uses unicast to communicate with a DNS server. In a Zeroconf network there is no dedicated name server, so queries in .local. namespace use multicast. The process selecting a name to a host follows the same pattern than selecting an IP address. At first a host selects a name for itself, probes for it and if no conflict occurs, the host announces for the name. (Cheshire & Steinberg 2005, pp. 39 - 42)

Auto naming was first introduced in the AppleTalk network protocol suite as a part of zero configuration in 1980s. The protocol used most widely nowadays is called mDNS (multicast DNS) and is developed by Apple. Multicast DNS is based on an IETF draft titled Multicast Domain Name Service written by Woodcock & Manning (2000) and Apple's own preceding protocols. Multicast DNS will be covered in detail in Chapter 5. Microsoft also made its own version, Link-local Multicast Name Resolution, LLMNR. It is published as informational RFC 4795, but has not gained substantial status like mDNS. (Cheshire 2012) LLMNR will not be covered any further.

2.1.3 Service discovery

Now we have gone through the Zeroconf tools for the IP address and name acquisition. Hosts can find each other using IP addresses or name corresponding to the IP address. Some might think that from the name of the host we can deduce what services it is capable to offer, at least to some extent. This is not exactly the truth. For example a host called HomePC may offer file server and web server services although its name does not indicate it in any way. Service discovery (SD) is used by hosts to discover the required services. There are a few mechanisms developed for service discovery. DNS has the extension called the DNS-SD. Other mechanisms covered here are Service Location Protocol (SLP) and Simple Service Discovery Protocol (SSDP).

All service discovery protocols discussed here use multicast to search and announce services. Such multicast addresses are divided into scopes that reflect the size of the network. These scope specific IPv6 multicast addresses begin with ff0x::, where x is a scope specific value depending on the scope in question. Table 2.1 presents scope specific values of multicast addresses.

Table 2.1. *Scope specific values for multicast addresses (modified from Hinden & Deering 2006, p. 14)*

Value	**Scope**	**Value**	**Scope**
0	reserved	8	Organization-Local
1	Interface-Local	9	unassigned
2	Link-Local	A	unassigned
3	reserved	B	unassigned
4	Admin-Local	C	unassigned
5	Site-Local	D	unassigned
6	unassigned	E	Global
7	unassigned	F	reserved

Next sections describe the three service discovery protocols and assume that a host has already acquired an IP address and a name. Scope specific values can be consulted from the table above.

DNS-SD

DNS-SD allows hosts to look for a service using a standard DNS queries by indicating the type of service and the domain from which it is looking for. This action is called the service instance enumeration and it can be understood as browsing the instances providing the requested services. DNS-SD uses DNS PTR or pointer records to get the information. In the conventional DNS PTR record is used for reverse mapping of a name to an IP address. In DNS-SD it is used to list services for a specified type in a specified domain. A host searching for a service looks for DNS PTR records for a specified service in a domain specified in the lookup. The format of the lookup is "<Service>.<Domain>". The answer format providing the instance name or names of the requested service is:

Service Instance Name = <Instance>.<Service>.<Domain>.

Depending on number of the PTR records matching the lookup the requester receives zero or more service instance name strings. According to Cheshire & Krochmal (2013b) the instance part of the service instance name string should be in user-friendly format, since it is displayed to users to help them to choose the most suitable instance. Prior to starting a service and announcing an instance user should be able to edit the default instance name given by the software for the same reason. (Cheshire & Krochmal 2013b, pp. 6-7, 9)

After the query host knows, which instances can provide the service it needs. Before a host can contact an instance from the list it has gathered from the received PTR records, it has to query for the SRV and TXT records for the name to get more information about an instance, such as IP address and port. (Cheshire & Krochmal 2013b, p. 10)

PTR record holds only the essential information to contact the host offering the service, mostly due to syntax rules of the PTR record. To be able to pass the additional information to requester DNS-SD uses TXT or text record. For every DNS PTR record used for DNS-SD the corresponding TXT record must exist, although it may contain only a single zero byte if there is no additional information available. If TXT record should not be mandatory, requester could not be sure, which case is correct when not receiving it, the TXT record value is zero or it did not get through. (Cheshire & Krochmal 2013b, p. 11)

DNS-SD described by Cheshire & Krochmal (2013b) works only in a single local-link. Cheshire (2013) describes a hybrid proxy function to overcome this limitation. It may reside in a router that connects different local-links physically or logically. Same site may have multiple proxies depending on the network topology. Hybrid proxy uses both unicast and multicast DNS to make queries and reply to them. For example, if it receives a unicast DNS query for a service, it sends out a multicast DNS query to the local-link segment or segments it is connected to. Upon receiving a multicast reply or replies, proxy gathers information valid for the original requester and passes them on to the requester using unicast DNS reply. Some parts of the specification are already implemented in current software releases, although the specification does not yet have IETF standard status. (Cheshire 2013)

Security aspects of DNS-SD are the same as in conventional and multicast DNS. Conventional DNS and multicast DNS are discussed further in later chapters.

SSDP

Simple Service Discovery Protocol or SSDP is another protocol developed to advertise services in an IP based network. It was designed for home or small office networks to help user to find services within the network. SSDP uses multicast, so users should not need to make any static configurations to be able to use it or the services in the network. IPv4 multicast address allocated to SSDP is 239.255.255.250 and UDP port is 1900. Corresponding IPv6 address is FF0X::C, where X depends on the scope in question. (Goland et al. 1999)

SSDP uses HTTP 1.1 specification partly. SSDP message format is equal to HTTP 1.1 specification, but in the transport layer it uses UDP instead of TCP. SSDP has three types of messages: NOTIFY, M-SEARCH and search response. NOTIFY messages are divided further into three different types as well. When a device is connected to a network, it must send the NOTIFY messages with ssdp:alive attribute to advertise its services. Messages are repeated a few times to be sure that every host connected to the network gets the message. Alive message has an expiration time on its header to ensure that advertisement is eventually expired from other hosts' memory in case they do not receive byebye message or the advertising host is suddenly removed from the network. When a host is removed from the network in a controlled manner, it should send the NOTIFY ssdp:byebye message to inform other hosts that it and its services will be removed from the network. The NOTIFY ssdp:update messages are sent to inform the network that a multi-homed host has a new interface connected to the network or it has removed one of its existing interfaces from the network. (Presser et al. 2008)

The M-SEARCH message is used by a host to search for a host or service from the network. The M-SEARCH message is sent with the ssdp:discover attribute using multicast. The M-SEARCH messages may also be sent using unicast when host's IP address is known. Unicast messages are used, for example, when it is needed to check that the host and its service a still reachable. A host must respond to the search request when the request matches to the host itself or services advertised by it. Response message format is very alike to the NOTIFY ssdp:alive message. The main difference is that NOTIFY ssdp:alive messages use multicast while M-SEARCH messages use unicast. (Presser et al. 2008)

SSDP was original described in an IETF draft document titled Simple Service Discovery Protocol/1.0 Operating without an Arbiter (Goland et al. 1999). The draft has expired since and the current specification of the SSDP is included in the UPnP (Universal Plug and Play) standard documentation by the UPnP Forum. Both UPnP and DLNA (Digital Living Network Alliance) use the SSDP to find services in the network they are used in.

SLP

SLP (Service Location Protocol) is another protocol to discover and advertise services to other hosts connected to the same local network. As the two previously presented protocols, DNS-SD and SSDP, SLP is also developed to function in an IP network. SLP uses UDP by default, but can switch to TCP if a message does not fit into a single UDP datagram. TCP/UDP port number used by SLP is 427. Unlike SSDP, SLP is designed to scale also to enterprise networks, but not to global ones, that may consist of hundreds of thousands of clients or tens of thousands of services. (Guttman et al. 1999)

SLP is based on three entities, User Agent (UA), Service Agent (SA) and Directory Agent (DA). User and service agents handle the communication on behalf of the client and server applications. Directory agent is a directory entity for the services that exists in the network. It listens to the network and collects service advertisements. Directory agents are not mandatory. They are merely used to provide scalability in larger networks by suppressing the multicast traffic sent by user and service agents. (Veizades et al. 1997; Guttman et al. 1999)

At first, when a user or service agent is connected to the network, it must search for the directory agent before sending any service requests. The information may be configured statically to the agent, but it should be considered as the least preferred manner. Better manner to resolve the directory agents' IP addresses are using DHCP, sending a service request query to the directory agent discovery multicast address 224.0.1.35 or FF0X::123 or listen to the service location general multicast address 224.0.1.22 or FF0X::116 for the directory agent advertisements messages. See the list of scope specific values in parent section 2.1.3. (Veizades et al. 1997)

If a service agent discovers a directory agent by some means, it must send register messages (SrvReg) containing all services it advertises to the directory agent. The directory agent replies with the acknowledge message (SrvAck) that it has received the register messages. After this service agent must periodically refresh the registration before the lifetime of the previous registration expires. (Veizades et al. 1997)

User agent acts in two different manners, depending on whether it knows a directory agent or not. If a user agent knows about the directory agent, it only has to communicate with it via unicast. If the last directory agent disappears from the network or there is no directory agent in the network, user agent uses multicast to communicate directly with a service agent. There is a situation when the directory agent exists, but the user agent still multicasts requests directly to service agents. This may be due to a configuration or light-weight design of the user agent. In this situation a service agent replies directly to the user agent despite the existence of the directory agent. (Veizades et al. 1997)

Messages sent between agents may be secured using so called protected scopes. Scope is a set of services that typically form a logical administrative group. Services in protected scopes have been signed using the public key infrastructure (PKI), so the use of these protected scopes can be limited to the authorized agents only. Signing services also shows user agents that service provided by a service agent or a directory agent is what it is supposed to be. Securing messages with PKI is not easy in minimum configuration networks. To be able to use protected scopes a service agent has to have a CA (Certificate Authority) signed certificate which allows it to sign services it advertises. Public part of the signing keys needs to be available for directory and user Agents. User agents are usually assumed to have a minimal or no pre-configuration when they are connected to network, so to be able to find and use these secured services they have to get the public keys of the service signed. Using protected scopes enhances the security of the network. A rogue client cannot discover or use services that are protected by the PKI, even though it manages to configure an IP and a name for itself. (Veizades et al. 1997)

Previous paragraphs described every type of an agent as a separate entity. In a live network these entities may reside in the same host, in principle. In practice user and service agents may reside in the same host. Directory agent may reside with either of the two other agents or by itself.

3 NETWORK ARCHITECTURE

Currently a typical home network is very simple. The figure 3.1 below illustrates an example topology. It consists of a router or a bridge that routes or bridges the traffic between home network devices and the Internet. Router usually receives only a single IPv4 address from an ISP and uses NAT (Network Address Translation) and private IPv4 addresses in the home network. In bridge mode, home network devices typically obtain a public IPv4 address from an ISP while the bridge only passes the network layer traffic between home network and the Internet acting more like a switch. At the moment only few devices other than computers and other networking devices at home have the ability to connect to the network. This fact also keeps the network size quite small both physically and logically.

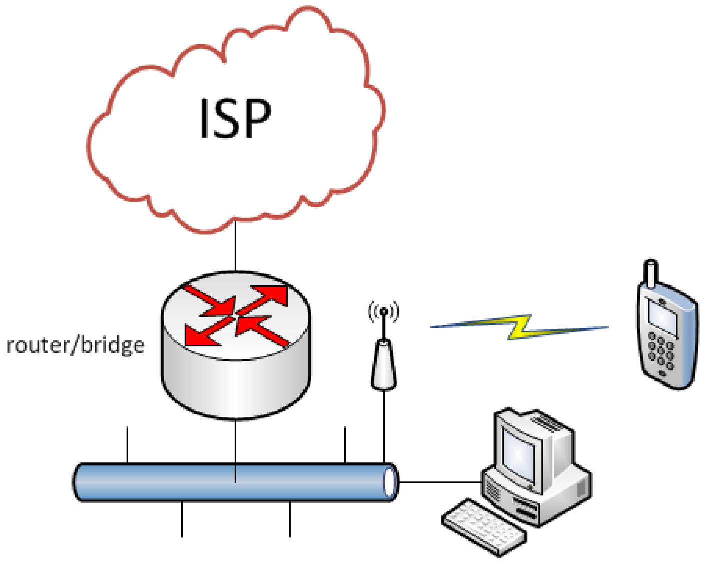


Figure 3.1. *Current typical home network*

In the near future home networks have to migrate to IPv6. This is because of the ISP will run out of IPv4 addresses reserved for them. Already many household equipment and building service technology systems are developed to be network enabled. They use it to update themselves for a better performance and user experience. Devices becoming network enabled and equipped with more and more features will lead to exponential growth of devices connected to the home network. Some devices may also be too old to be updated to support IPv6.

3.1 The future home network

Home network size will grow as more and more household equipment become network enabled. Some of these systems to be connected to network may even be so critical, that the demand for reliability forces home customers to consider and, in some cases, purchase a redundant connection and devices. Rational way to setup a redundant connection to the Internet is to purchase another connection from a different ISP. To be able to accomplish these requirements of redundancy and availability home networks become more and more like some IPv4 enterprise networks already are today. Homenets may soon have a wired primary connection and a wireless backup connection to the Internet. The word homenet is used as the synonym for the word home network in this chapter and the following chapters.

Multi-homing is a technique to introduce a redundant connection to the network outside the homenet. In IPv4 homenet there has not been any best practice for it. One reason is the use of private addressing and NAT due to a limited address space of IPv4. IPv6 address space on the contrary has enough addresses for every device on Earth. Every host in homenet can have its own globally unique IPv6 address from each ISP the homenet is connected to.

3.1.1 Multi-addressing of an IPv6 host

IPv6 protocol introduces multi-addressing. Even in a simple homenet connected to the Internet each device has a globally unique IPv6 address and an IPv6 link local address. In some cases, if network is dual-stack a host also has an IPv4 address. In the IPv6 homenet, it should be considered the norm for devices to be multi-addressed. (Chown et al. 2013, p. 8)

There are several reasons for multi-addressing. One is the same as in IPv4 Zeroconf, discussed in the previous chapter. Homenet devices should have a local scope address alongside with a globally unique address in case of an outage on the ISP connection. Thus the local network is operational during the outage.

Another reason is multi-homing. Homenet reachability can be increased by introducing a redundant connection to another ISP. To be able to use that connection, devices in the homenet must acquire another globally unique IPv6 address from another ISP. RFC 6724 written by Thaler et al. (2012), titled Default Address Selection for Internet Protocol Version 6 (IPv6), defines the rules for source and destination address selection for IPv6. In the case of dual-stack, the document advises to use the corresponding IP version for the source address as the destination address is. Also if the destination, for example, a web server, has the both IPv4 and IPv6 global scope addresses, but the source host has one global and another non-global address, for example RFC 1918 defined private IPv4 address or link-local IPv6 address, the host should prefer the IP version, from which it has a valid global scope source address. To be able to optimize the communication between hosts both, source and destination, addresses must be selected properly.

Address selection algorithm described by Thaler et al. (2012) applies only to IPv6 source addresses. For destination address selection, IPv4 addresses must be presented as IPv4-mapped addresses. A simple manner for the selection is to think that only the source host needs to make the selection and the destination host can use the same addresses only switching them vice versa when sending the reply. This schema is valid in the unicast communication, but in the multicast communication the destination host has to select the source address for the reply, since in the packet it received, the destination address was multicast address. Although the rules for both the source and the destination address selection algorithm differ, the combined effect is to prefer the source and destination address pairs that have an equal scope or type and to prefer smaller scopes over larger ones for the destination address. The preferred source addresses should be used over the deprecated source addresses meaning that expiring address should never be used for communication when there is a newer, valid address present. Transitional addresses should be avoided when the native addresses are available. This rule implies to prefer the native IPv6 addresses over the IPv4-mapped addresses. The last, discriminating rule is to prefer address pairs that have the longest possible common prefix.

From the homenet point of view the above rules mean that in communications within the homenet the local scope addresses are to be used, but when communicating to the Internet the global scope address should be used.

3.1.2 Homenet topologies

Chown et al. (2013, pp. 14-17) describe three different topologies for the future homenet. The first one is closest to the current IPv4 homenet topology. The second and the third are multi-homed. In all topologies home router between the homenet and an ISP network is called the CER (Customer Edge Router). It may be controlled by the ISP or the customer.

Single ISP

The first topology has only one connection to ISP via single CER. The network may have an internal router or routers. Addressing scheme of the network is quite simple. The CER acquires a globally unique prefix from the ISP and distributes it to devices inside homenet. If an internal router exists, configuring it may require hands-on activity from the network administrator. The figure below represents an example topology.

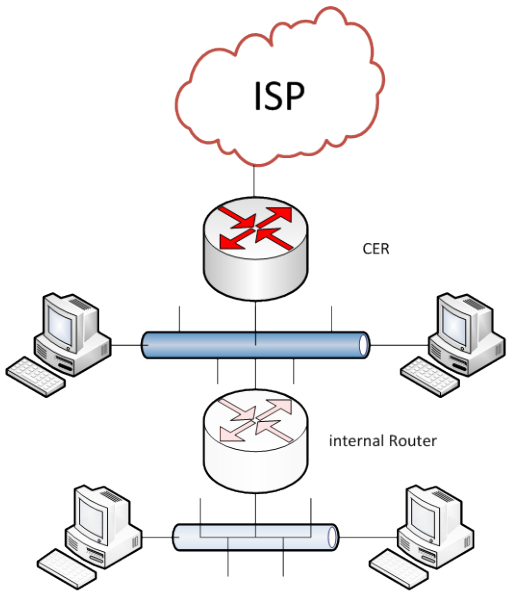


Figure 3.2. *Network diagram - Single ISP*

The next two topologies are multi-homed. In uncommon cases where multi-homing is used in IPv4 homenets, the most used solution is to isolate the homenet to multiple physical or logical networks. Homenet devices are divided into both networks. If either ISP connection fails, another connection and that part of the network should remain operational.

Multi-homing

In the IPv6 multi-homing topology the homenet is connected to two different ISPs with separate physical connections. The figures below present two types of multi-homing topologies described by Chown et al. (2013, pp. 14-17). The figures are simplified from the originals. The first figure represents a topology with a single CER connected to two ISPs or to the same ISP with two different connections.

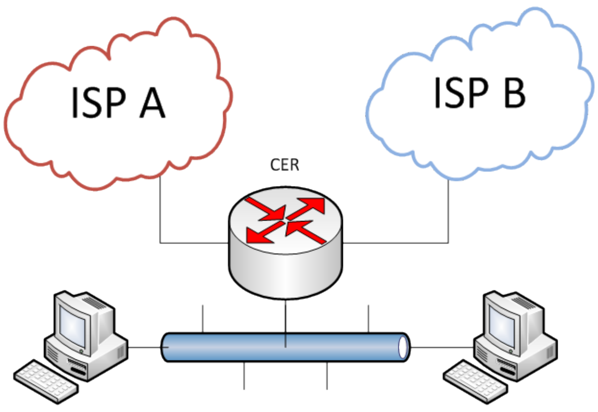


Figure 3.3. *Network diagram - Multi-homing - Single CER*

In this scenario multi-homing may be accomplished through source routing. CER forwards packets to the correct ISP connection depending on the source address of the

IP packet. The disadvantage is the single CER. If CER fails, both ISP connections are lost.

The next figure introduces redundant CER. It isolates ISP connections to separate CERs, thus outage of a CER does not isolate the entire homenet from the Internet.

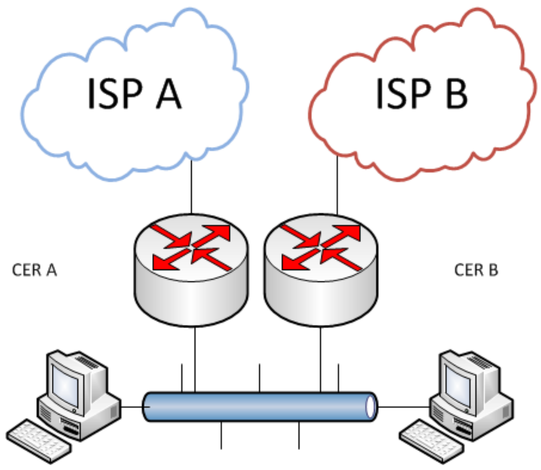


Figure 3.4. *Network diagram - Multi-homing - Two CERs*

With the two CERs, the network layer complexity rises. In this scenario the homenet has to be able to route packets to the proper egress. If this fails, ISP will not accept and forward the traffic due to the RFC 2827, which states that a service provider must drop packets coming from the subscriber, if the source address of the packets is not from the prefix that the ISP assigned to the subscriber (Ferguson et al. 2000, pp. 5-6). Chown et al. (2013, p. 18) recommend that packets should be routed to the correct ISP connection with the most efficient manner while avoiding to drop any packets.

Both multi-homing topologies may have one or more internal routers in homenet, which may require additional configuration by the administrator. Multi-homing challenge is broad and there are several solutions for it. Chown et al. (2013, p. 19) conclude the subject by saying that homenet architecture should minimize the complexity of multi-homing support as far as possible.

3.1.3 Eliminating NAT

IPv6 address space is vast compared to IPv4 address space. It enables end-to-end communication with IPv6. Every host in every network can possess a globally unique IPv6 address. This is an opportunity as well as a challenge to the networks to be migrated to IPv6.

Chown et al. (2013, pp. 17 – 18) state that most home networks that implement IPv6 in the near future will still use IPv4 in parallel. NAT used in IPv4 networks will be replaced by pure routing in IPv6. IPv6 networks using the same topology as IPv4 networks should work better and be independent from IPv4. The document mentions that bridging may still be used in IPv6 homenets, but for various reasons it should not be preferred.

Removing NAT and thus enabling end-to-end communication makes IPv6 an interesting opportunity. The key statements in Chown et al. (2013) document, covering the future IPv6 homenet architecture concerning addressability and reachability is that network addresses must not be altered in transit. This allows hosts to be contacted directly anywhere from the Internet, without any administrative intervention. This creates a challenge, since NAT is still the only thing separating the majority of the IPv4 homenets from the unsecure Internet, thus providing an apparent security. When NAT is removed, hosts in homenets become completely visible to the Internet, exposing them to the whole range of hazards existing in the Internet. This is mainly due to the fact that hosts in homenets have poor level of security, especially hosts running legacy operating systems. Security means here firewall and anti-virus software, but also out-of-date software having known security flaws. Luckily operating systems nowadays come at least with built-in firewall software, which provides some security. Operating systems and software stay automatically up-to-date, since the Internet access is available for majority of the people at an affordable price.

Although computers, smart phones and tablets may have sufficient security, there are still devices connected to homenet that need external security against the Internet. Devices such as printers, home surveillance system and so on should be isolated from the Internet through a boundary between homenet and the Internet. Van de Velde et al. (2007) present basic security features to protect the local IPv6 network. The document discusses widely the use of NAT in IPv4 networks and presents features provided by IPv6 to replace it in IPv6 networks. Main idea is to introduce a firewall service in the gateway between the homenet and the Internet. Firewall should have static filters to check all packets to and from the Internet for basic sanity like spoof and have stateful filters for flows originating from homenet to be able to distinct and reject flows originating from the Internet. Gateway should also implement some type of logging for network and transport layer information in order to track communication of applications. Services provided for the intranet by the gateway, such as DHCPv6 and DNS resolver, must not be available to the Internet. There should also be a possibility to modify the default configuration of firewall and other options.

From Van de Velde et al. (2007) specification can be seen that it is written more from corporate point of view, since its approach is more natural to corporate administrators than ordinary home users, thus denying all traffic unless it is not explicitly allowed.

Woodyatt’s (2011) point of view is a bit different. He presents fifty recommendations for simple security capabilities for the homenet gateway. The basic recommendations are the same, including having stateless filters for invalid headers, spoofed packets and so on. Stateful filters should also exist for filtering unwanted inbound traffic from the Internet towards homenet. But the second last recommendation says that a transparent mode may be a default mode for homenet gateways. In transparent mode the gateway does not use the IPv6 simple security capabilities, but should forward all traffic transparently. If transparent mode is used as default mode, all homenet hosts are revealed to the Internet. Therefore their security needs to be revised with extra care.

Both, Van de Velde et al. (2007) and Woodyatt (2011) state that dynamically opening holes to the gateway firewall for inbound traffic from unknown source addresses need further study. In IPv4 networks with NAT this has been accomplished using UPnP or a similar protocol.

4 DNS

DNS is a hierarchical distributed naming system for computers, services and other resources connected to the Internet. The most important task of the DNS is to translate domain names to IP addresses in order users to access the resources they want to. The chapter deals with the relevant parts of the DNS for this thesis.

In this chapter, at first the basic building blocks of the DNS, the resource records, are introduced briefly, followed by the types of servers implementing the DNS. Reverse DNS, DNS message types and zone transfer mechanisms are discussed. The last sections of the chapter concentrate to DNS security, covering security of DNS messages and the whole DNS system.

4.1 DNS in operation

DNS zone presents a set of names under the same authority. A zone may comprise multiple parent and child domains. For example a parent zone example.com and its child zones ftp.example.com and homenet.users.example.com. These child zones or sub-domains of a parent domain may also be delegated into a separate authority, depending on the parent domain administrator’s decision. The term delegation is used when a parent domain administrator hands over the responsibility of a sub-domain or child zone to another authority. Terms child and parent authority are used to distinguish the relationship between the two instances. (Gudmundsson & Koch 2007, p. 5)

4.1.1 Resource Records

Data in a zone file is represented as Resource Records (RR), which is the basic data element of the DNS. Resource record contains the data associated to the domain in question. All resource records have the same top level data format shown in the figure 4.1 below. The whole DNS is based on these records. All data stored to the DNS is in resource records.

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                                               /
/                      NAME                     /
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     CLASS                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TTL                      |
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                   RDLENGTH                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--|
/                     RDATA                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 4.1. *DNS Resource Record basic data format (Mockapetris 1987, p. 11)*

Explanations of the fields:

- NAME: describes domain name of the node to which this resource record pertains
- TYPE: contains the type of resource record in numeric form, e.g. 1 = A, 15 = MX and 28 = AAAA
- CLASS: contains the class of resource record, usually IN, others, non-global, also exist
- TTL: time in seconds, that resource record is valid
- RDLENGTH: length of RDATA field
- RDATA: additional data to describe the resource record, format of the field depends on TYPE and CLASS fields of the resource record. (Mockapetris 1987, p. 11-12; IANA 2013d)

Name field contains, for instance in case of type A or AAAA record, the host portion of the FQDN (Fully Qualified Domain Name). Depending on the type of record the name field may contain various types of names.

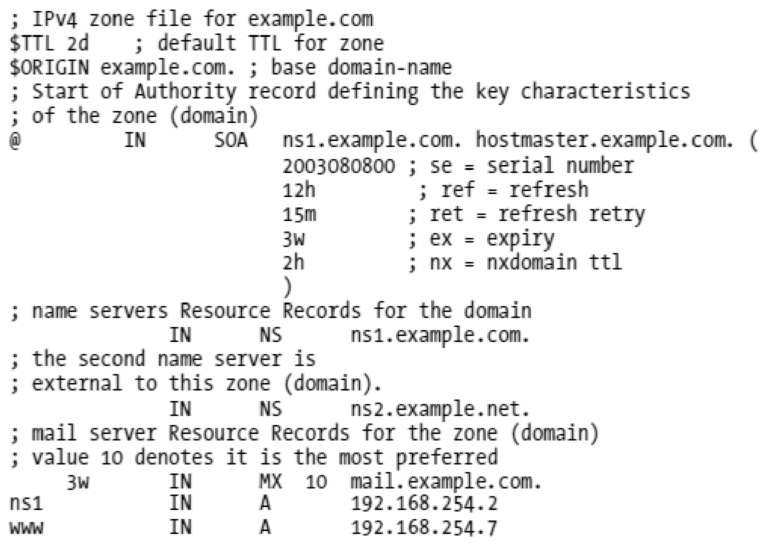
At the moment there are about 80 different types of resource records. Type field tells which type of record is in question. Type numbers 1 and 28 and their names A and AAAA correspond to IPv4 and IPv6 address records. Number 15 or MX is a record type for mail exchange. (IANA 2013d)

Class field indicates the class of the record. At the moment there is only one global class IN (Internet). Also CH (CHaos) and HS (HeSiod) classes exist, but are only in experimental use. (IANA 2013d)

TTL or Time To Live field reveals how long in seconds the record is valid. After the time in the TTL field has elapsed and update has not been received, the resource record must be removed.

RDLENGTH field reveals the length of the following RDATA field. This is important because the length of the RDATA varies depending on information in it. All other fields, including the RDLENGTH field, are fixed size fields.

RDATA field carries the actual information of the resource record. It may contain the name of the DNS server for the domain or a public key of a host, for example. (IANA 2013d) The next figure represents an example zone file in format used by the BIND (Berkeley Internet Name Domain).



```
; IPv4 zone file for example.com
$TTL 2d    ; default TTL for zone
$ORIGIN example.com. ; base domain-name
; Start of Authority record defining the key characteristics
; of the zone (domain)
@         IN      SOA   ns1.example.com. hostmaster.example.com. (
                        2003080800 ; se = serial number
                        12h         ; ref = refresh
                        15m        ; ret = refresh retry
                        3w         ; ex = expiry
                        2h         ; nx = nxdomain ttl
                        )
; name servers Resource Records for the domain
              IN      NS      ns1.example.com.
; the second name server is
; external to this zone (domain).
              IN      NS      ns2.example.net.
; mail server Resource Records for the zone (domain)
; value 10 denotes it is the most preferred
     3w       IN      MX  10  mail.example.com.
ns1           IN      A       192.168.254.2
www           IN      A       192.168.254.7
```

Figure 4.2. *An example of a zone file (modified from Aitchison 2011, p. 27)*

The SOA (Start of Authority) is the most important record type when creating a zone. It holds vital information of the zone or domain. In the example the ns1.example.com is the authoritative name server for the zone. Depending on the DNS implementation it may be master or a slave server. Next string hostmaster.example.com is actually an email address of the zone administrator. The series of numbers in the next row is a serial number for the zone. Every time a change occurs in a zone the number is increased, which in turn tells slaves that they have to transfer updated zone information from the master server. Other time values are related to how long the zone information is valid and how often it should be refreshed.

Resource record can be grouped into RRsets. RRset is a group of resource records that have the same NAME, TYPE and CLASS fields. Other fields may differ. If also DATA field is the same for the two resource records, then duplicate RR exists and should be suppressed. (Davidowicz 1999) In the example above, the two NS records form an RRset. Terms such as authoritative name server and master server used in preceding section will be explained further in the following sections.

4.1.2 Servers and resolvers

Resource records are the basic data elements of the DNS regarding the information stored to DNS. The actual devices holding and distributing the DNS information can be divided also into two main elements. They are DNS servers and resolvers. The servers holding the DNS information for a domain are called authoritative servers. They possess the zone file, which holds the information about certain administratively domain. DNS rules require that every domain has to have at least two authoritative servers, one master and at least one slave. This requirement is for redundancy reasons. When one server goes down, the other one continues to answer to queries concerning that domain. Zone data from the master is dynamically transferred to slaves via zone transfer. (Gudmundsson & Koch 2007, p. 7; Albitz & Cricket 2006, p. 51)

DNS resolvers can be divided further into two, stub and recursive resolvers. Stub resolver just asks questions and waits for complete answers. Stub resolver is a DNS client running on a terminal, such as computer or smartphone. Stub resolver sends a recursive query to the name server configured to it, with recursive bit on. A DNS server can be configured to act as a recursive resolver, but does not necessarily have to accept recursive queries. Upon receiving recursive DNS query from a stub resolver a recursive resolver starts to resolve the query by sending iterative queries level by level starting from the root level and working down to the domain tree as long as it gets the IP address of the host in question. Once it has the IP address of the host queried, it replies to the requester, the stub resolver, who sent the query. (Gudmundsson & Koch 2007, p. 7; Albitz & Cricket 2006, p. 52-57) Below is a conceptual figure about the name query process described above. The computer on the left is a stub resolver querying for www.dnsop.org and the server in the middle is the recursive resolver.

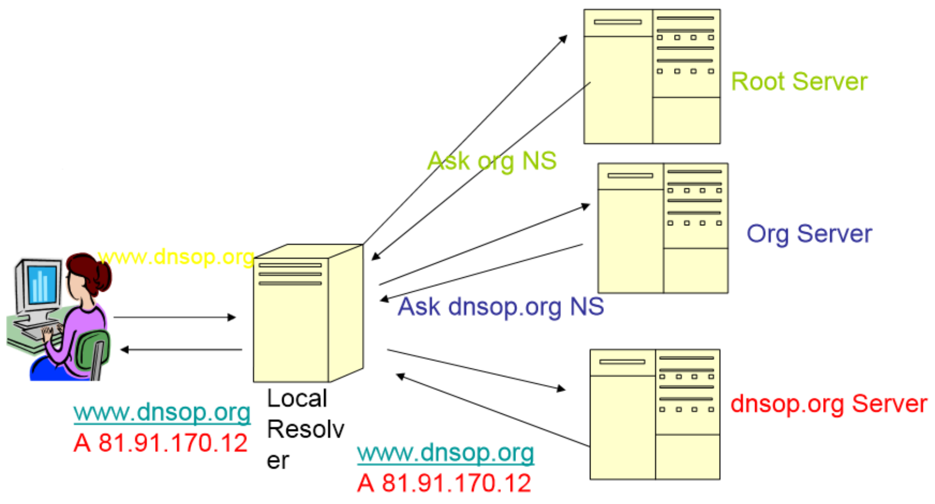


Figure 4.3. *Recursive DNS query (Gudmundsson & Koch 2007, p. 11)*

Starting every new query by asking root servers about the top level domain information and continuing down level by level creates large amounts of DNS traffic. To reduce this a caching resolvers are used. Caching resolver is a recursive resolver that stores answers to preceding queries and uses the acquired information for subsequent queries. Sometimes resolver may have a complete answer stored and it can reply immediately to the query. Caching resolver may also have only partial information, so it has to make some additional queries before it can respond to the querier. Again information got, is stored for later use. (Gudmundsson & Koch 2007, p. 7; Albitz & Cricket 2006, p. 59-60)

Previously described DNS elements may reside in the same host. For example, a host may act as authoritative server and as recursive resolver simultaneously. Authoritative server provides authoritative answers only for data it holds in its own zone file. Acting also as a recursive resolver, it provides non-authoritative answers to queries, such as the recursive resolver in the preceding figure.

4.1.3 DNS server topologies

DNS server for a zone can implement various topologies. There must always be at least two authoritative servers for each zone. This requirement is for redundancy purposes as discussed earlier.

The most common DNS topology is the master-slave topology represented in the figure 4.4 below. The master server is responsible of the zone. The information is always updated to the master server zone file. Slave servers retrieve the updated information using zone transfer protocols or some other methods discussed in Section 4.4. Both master and slave can receive queries for the zone and may reply to them. (Aitchison 2011, p. 20)

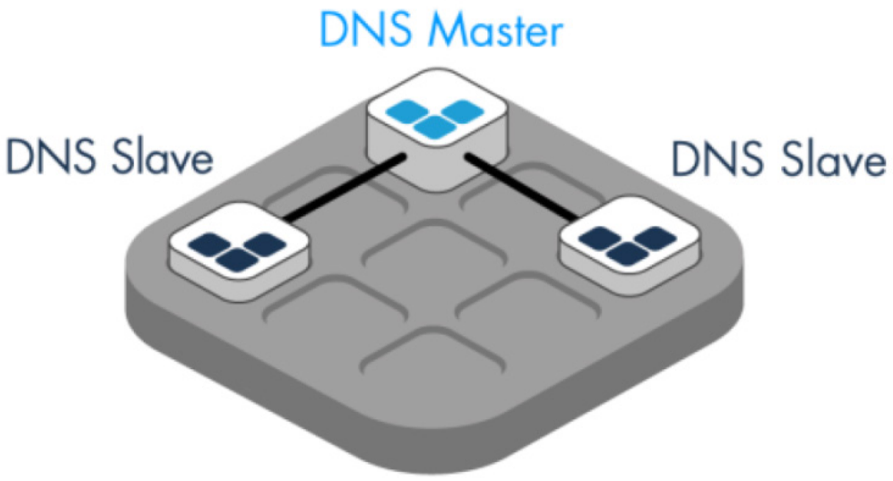


Figure 4.4. *Master - slave topology (modified from EfficientIP 2012, p. 4)*

The other common topology is the hidden or stealth master topology. Here the master server is hidden from the rest of the network the zone is serving. It is accomplished using a certain kind of zone configuration. Using the figure 4.2 as an example, the zone file in the hidden master topology will not contain any reference to the master server. The SOA record refers to a slave server as do the NS, A and AAAA records as well. The only reference to the master is in a separate file in the slave server. The file contains the IP address of the master server. This way a slave can transfer zone information from the master server. As mentioned also in the figure below, some sources refer the slave in the SOA record to as a pseudo master.

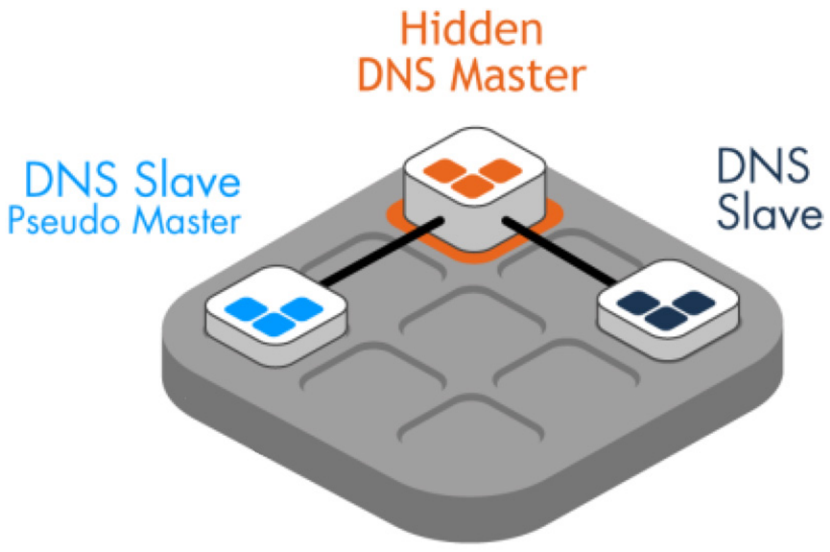


Figure 4.5. *Hidden master topology (modified from EfficientIP 2012, p. 16)*

Other topologies also exist, but they are usually mixture or modification of the preceding topologies.

4.2 Reverse DNS

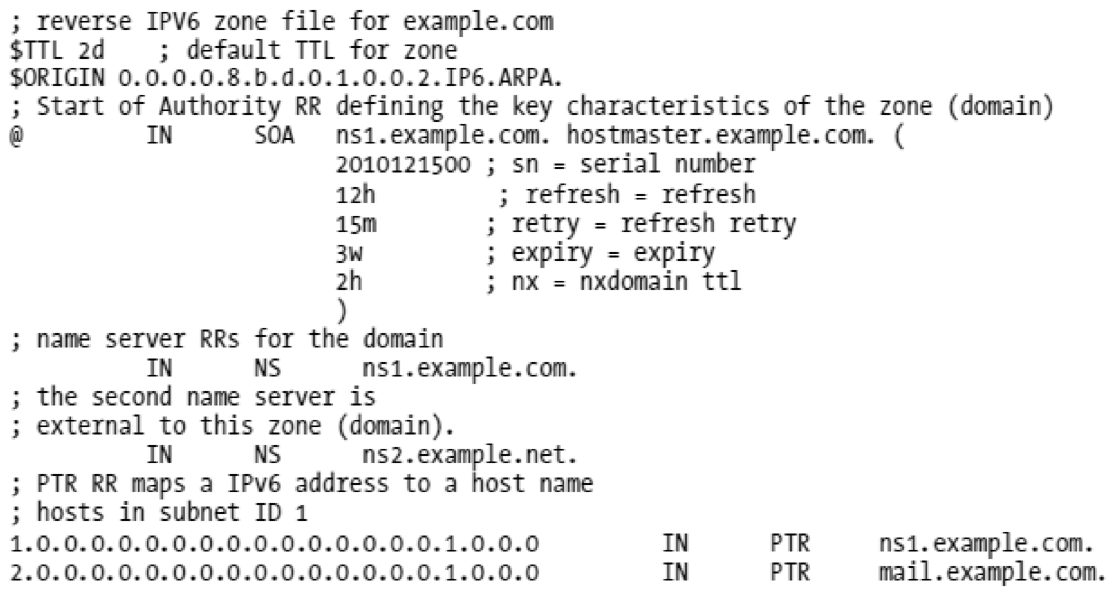
The primary use of the DNS is of course to map a name to an IP address, but sometimes it needs to be done in reverse. Reverse DNS maps IP addresses to names. A reverse zone file contains static entries for each host that requires reverse information. In reverse zone file IP addresses are in reverse byte order with a suffix .in-addr.arpa. for IPv4. IPv6 address suffix is .ip6.arpa., respectively. For example, IPv4 address 192.168.254.4 translates to 254.168.192.in-addr.arpa. The last number, the host portion is omitted, since it is defined in a separate row of the zone file. The host portion of IP address is expressed in the PTR or pointer resource records. In conventional DNS the PTR record is used only to map an IP address to a particular host in a domain. In multicast DNS it has different purpose. (Aitchison 2011, p. 47-49, 52) Below is an example of a reverse zone file.

```
; simple reverse mapping zone file for example.com
$TTL 2d    ; default TTL for zone
$ORIGIN 254.168.192.IN-ADDR.ARPA.
; Start of Authority record defining the key characteristics of the zone (domain)
@         IN      SOA   ns1.example.com. hostmaster.example.com. (
                        2003080800 ; sn = serial number
                        12h         ; refresh
                        15m        ; retry
                        3w         ; expiry
                        2h         ; nxdomain ttl
                        )
; name servers Resource Records for the domain
          IN      NS      ns1.example.com.
; the second name server is
; external to this zone (domain).
          IN      NS      ns2.example.net.
; PTR RR maps an IPv4 address to a host name
2         IN      PTR     ns1.example.com.
```

Figure 4.6. *An example of an IPv4 reverse zone file (modified from Aitchison 2011, p. 49)*

The most important parts of the zone file regarding reverse resolution are third row, starting "$ORIGIN…" and last row starting with "2" and "4". The third row expresses the IP subnet that the zone file manages. Network part of the C-class subnet can be extracted from that row, which is 192.168.254.x. Last octet, the host portion of the prefix is extracted from the last rows of the zone file. So if a query for 192.168.254.4 is received, DNS server will answer "mail.example.com". The reason for the IP addresses to appear in reverse octet order is mainly administrational. By reversing the octets the responsibility to maintain reverse DNS information can be easily delegated to the authority responsible of the subnet. Keeping IP addresses in normal octet order would make the delegation practically impossible. (Albitz & Cricket 2006, p. 58-59)

Syntax of an IPv6 reverse record is the same, of course. In IPv6, reverse record has to be constructed with a nibble precision. A nibble is a half of a byte, a 4 bit unit. Otherwise the IPv6 prefix has to be divided into several reverse records. When this is required, the prefix is first extended to next more specific nibble boundary prefix. So many extended prefixes are used, as is needed to cover the original prefix. Reverse zone files are then constructed, one for each extended prefix. PTR resource records are constructed in the same manner as in IPv4. Below is an example of an IPv6 reverse zone file. IPv6 addresses may create a long name to the PTR records. They can be shortened by moving some of the host address bits to the zone domain name. (Aitchison 2011, p. 89-90)



```
; reverse IPV6 zone file for example.com
$TTL 2d    ; default TTL for zone
$ORIGIN 0.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA.
; Start of Authority RR defining the key characteristics of the zone (domain)
@         IN      SOA   ns1.example.com. hostmaster.example.com. (
                        2010121500 ; sn = serial number
                        12h         ; refresh = refresh
                        15m        ; retry = refresh retry
                        3w         ; expiry = expiry
                        2h         ; nx = nxdomain ttl
                        )
; name server RRs for the domain
          IN      NS      ns1.example.com.
; the second name server is
; external to this zone (domain).
          IN      NS      ns2.example.net.
; PTR RR maps a IPv6 address to a host name
; hosts in subnet ID 1
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0         IN      PTR     ns1.example.com.
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0         IN      PTR     mail.example.com.
```

Figure 4.7. *An example of an IPv6 reverse zone file (modified from Aitchison 2011, p. 89-90)*

Reverse DNS is most commonly used to provide a simple security. For instance, modern email systems do not accept incoming emails, unless they can resolve a valid hostname for the sender's IP address. This is used to prevent spamming email servers to successfully send their spam, since they usually do not have valid reverse mappings. (Aitchison 2011, p. 47)

4.3 Messages

In the conventional DNS messages are sent using unicast transmission method. It is a one-to-one connection between a client and a server. Unicast messages are sent using UDP or TCP as the transport layer protocol. UDP and TCP port number assigned to the conventional DNS is 53. The transport layer protocol depends on the length of the message. The boundary is 512 bytes. In the original specification messages larger than 512 bytes are sent using TCP, while messages smaller and equal to 512 bytes use UDP. Damas et al. (2013) define an extension to enable also messages larger than 512 bytes sent using UDP. Normal queries do not exceed the boundary, but for example, a zone implementing DNSSEC (Domain Name System Security Extensions) and NSEC3 (NextSECure3) may send responses larger than 512 bytes. Previous term will be described further in Section 4.6.

DNS messages can be divided into three types – the queries, responses and updates. Format of the query and the response is the same, but the update message has slightly different format. Following sections present all three types.

4.3.1 Query and response

The query and the response are the original standard message types of the DNS. They were defined by Mockapetris (1987) in RFC 1035. Basic DNS communication is handled with these two messages. The figure below presents the format of standard DNS message.

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

Figure 4.8. *DNS message format (Mockapetris 1987, p. 25)*

In certain DNS messages some sections may be empty, but the header section is naturally always present. The header section specifies which of the remaining sections are present. (Mockapetris 1987, p. 25) The header includes the following fields:

- ID: identifies query and response pairs
- QR: identifies whether a message is query or response
- OPCODE: identifies the type of query, values may be from 0 to 15
- AA: identifies whether the response is an authoritative answer or not
- TC: identifies whether truncation is used or not
- RD: identifies whether the recursion is desired or not, used in query message
- RA: identifies whether the recursion is available or not, used in reply message
- Z: Reserved
- RCODE: Response code field, which may have value from 0 to 15
- QDCOUNT: specifies the number of entries in the question section
- ANCOUNT: specifies the number of RRs in the answer section
- NSCOUNT: specifies the number of RRs in the authority section
- ARCOUNT: specifies the number of RRs in additional section. (Mockapetris 1987, pp. 26-28)

The question section and the two following sections are named after their purpose. Question section is divided into three fields. QNAME field includes the domain name queried. QTYPE includes the type of resource record required, for instance A or MX. QCLASS field includes the type of class queried. Generally this is “IN”. The last three sections: answer, authority and additional all have the same format. They include a variable number of resource records. (Mockapetris 1987, pp. 28-29)

4.3.2 Dynamic DNS update

The DNS was originally designed to have a statically configured database, where updates were expected to happen infrequently. Updates were also expected to be made by externally editing the zone master file. This has not been possible for a long time for the most zone files, since the DNS needs to be updated quite frequently nowadays.

RFC 2136 (Vixie et al. 1997) defines the dynamic update feature to update DNS zone file. It uses a modification of the standard DNS message to add and delete RRs or RRsets. Update message cannot however modify every RR or RRset. An example is the SOA record. Since the DNS update message differs from the standard DNS message, it is presented separately.

Update message

Dynamic update uses modified DNS message format shown below. It differs from the standard DNS message by having a modified header section and different names and usage for the sections between the header and additional data section. (Vixie et al. 1997, p. 3)

```
+---------------------+
|        Header       |
+---------------------+
|         Zone        | specifies the zone to be updated
+---------------------+
|     Prerequisite    | RRs or RRsets which must (not) preexist
+---------------------+
|        Update       | RRs or RRsets to be added or deleted
+---------------------+
|   Additional Data   | additional data
+---------------------+
```

Figure 4.9. *DNS update message format (Vixie et al. 1997, p. 3)*

The header section of the DNS update message differs from the header defined in the RFC 1035 by naming and usage of the last four fields. The format of these fields is the same as the fields defined in standard DNS message. The header section of the DNS update message is shown in the figure below.

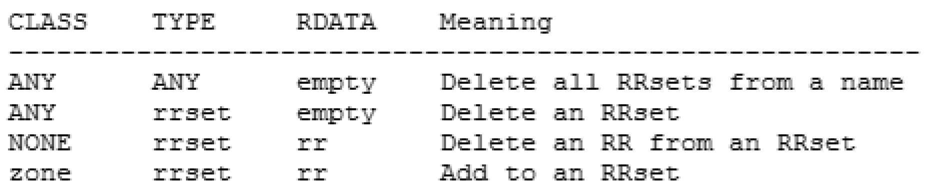
```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |          Z         |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ZOCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    PRCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    UPCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ADCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 4.10. *DNS update message header section (Vixie et al. 1997, p. 4)*

As seen from the figure above and compared to the standard DNS query and response message format fields between Opcode and RCODE are missing, since they are irrelevant in DNS update message. The field that defines a DNS message to be an update message is the Opcode field. With value of five (5), a DNS message is identified as an update message. The last four fields specify the number of RRs carried in the corresponding section. (Vixie et al. 1997, pp. 4-6)

Zone section expresses the zone of the resource record that is being updated. Zone section is allowed to contain exactly one zone, so only one zone can be updated at the time. Prerequisite section contains a set of RRset pre-requirements that must be met in order to successfully accomplish the update. Update section contains the data to be updated. RRs that are to be added or deleted are included in this section. Update section may contain four possible semantics: add or delete a resource record, delete an RRset or delete all RRsets. (Vixie et al. 1997, pp. 6-9) The following table illustrates the above semantics. It presents the metavalues used in update section.

Table 4.1. *Metavalues of the update section of the DNS update message. (Vixie et al. 1997, p. 15)*



CLASS	TYPE	RDATA	Meaning
ANY	ANY	empty	Delete all RRsets from a name
ANY	rrset	empty	Delete an RRset
NONE	rrset	rr	Delete an RR from an RRset
zone	rrset	rr	Add to an RRset

The additional data section contains resource records related to the update itself, or to new RR or RRs that will be added by the update. (Vixie et al. 1997, p. 10) Records related to the update message may be TSIG and SIG(0) resource records, that are used to authenticate the update message. Update message authentication is discussed later.

Server behavior

Upon receiving an update message, the master server will perform checks to every section to be able to determine, if the data carried by the update message can be used to update corresponding zone file. If the server does not find the RR or the RRset to match the update message at all, the message is silently ignored. Respectively, after prerequisites are checked to meet the current zone file, permission to update the zone file is checked. Permission may be based on a local policy or mechanism specified in secure DNS update protocol. After the update message is processed, server sends a response message, with correct response code to the requester. (Vixie et al. 1997, pp. 9-17)

4.4 Zone transfer mechanisms

There are three inband mechanisms used to maintain congruent information for a zone between DNS servers. The authoritative transfer (AXFR) defined in RFCs 1034 and 1035, the incremental zone transfer (IXFR) defined in the RFC 1995 and a mechanism to notify for zone changes (NOTIFY) defined in the RFC 1996. (Lewis & Hoenes 2010, p. 4-5) There are also other means to accomplish the same tasks, such as the rsync.

4.4.1 Zone transfer protocols

Lewis & Hoenes (2010) is used as a reference for AXFR, although it is not the original RFC defining the protocol. This is because the original RFCs 1034 and 1035 have the protocol definition scattered all over the documents and because the RFC 5936 presents the protocol as it is understood by the DNS community by the time of writing the referenced RFC.

AXFR was designed to request and transfer the contents of a zone from a DNS server to another. It uses the DNS message format with the help of some extensions. It should be noted that AXFR messages are sent using TCP instead of UDP. AXFR query is sent to the server whenever there is a reason for it. Query might be scheduled or triggered by an event such as NOTIFY message. Query fits in a single message while response may have to be divided into several messages, which it the main reason for the use of the TCP protocol. (Lewis & Hoenes 2010, p, 6-8)

AXFR response, including the whole zone data, is sent as a series of messages. The first message must start and the last message must end with the SOA resource record of the zone. This is the indication to the requester, that the complete zone information is received. Response messages can contain as many RRs as possible, but because of some legacy AXFR clients that may expect only a single RR per message a server may have a feature to include only one RR per response message. Server must copy the question section from the query message to the question section of the first response message. The question section of subsequent messages may be left empty. The queried zone information is copied to the answer section of the response message. (Lewis & Hoenes 2010, p. 11-14)

The NOTIFY messages are used by master servers to inform their slaves that zone information has changed. In original AXFR specification updates were not triggered, but scheduled. This increased the propagation time of the changes made to zone data. With NOTIFY message the propagation times will decrease, because master server can inform its slaves right after it has completed the change to its database. After a change is made the master updates the serial number of the zone and sends out a NOTIFY requests stating that there has been a change in resource records mentioned in the request message. Upon receiving a NOTIFY request a slave server should compare the data received against its zone data. Usually slave’s data is outdated and it sends a NOTIFY response to its master requesting updated zone data. Master then sends the requested data back to slave and so the slave's zone data is back up-to-date. NOTIFY messages may use either UDP or TCP. (Vixie 1996)

The weakness of the AXFR is that it sends the entire zone file. Usually only a fragment of the zone file is changed and should be sent. Ohta (1996) presents the incremental zone transfer (IXFR). IXFR only sends changed portions of the zone data. Master server keeps track of the serial numbers of the zone data and differences between them. Master may use previously presented NOFITY message or other means to start the IXFR process. When it receives an IXFR request with older serial number, it compares its data and sends only the changed portion of the zone data to slave. Another distinct difference between AXFR and IXFR is that IXFR may use either UDP or TCP, whereas AXFR is forced to use TCP. Zone transfer and NOTIFY messages may be secured using TSIG or SIG(0), which are discussed in Section 4.5.

4.4.2 Rsync

There are also other methods that can be used in place of the zone transfer protocols. As there are many others, rsync is used here as an example of those other methods. DNS administrators are using rsync utility to replicate the zone data between DNS servers. Rsync can be used to synchronize any type of files between hosts. It compares files on both hosts and transfers only the portions of files that have been changed. This way it reduces bandwidth used for file transfer. Rsync can be run over SSH connection to make transfer secure. (Davison, W. 2013) When using other means than the zone transfer protocols to maintain DNS zone integrity, extra work is always required. Careful consideration should be done before using some other means than the DNS protocol define.

In the Internet forums there are administrators for and against of using rsync instead of AXFR or IXFR. Some favor the use of rsync because of its security features, but admit that it needs a lot more configuring and scripting than AXFR or IXFR. AXFR or IXFR used with DNS NOTIFY message have their own supporters, who believe in simplicity.

4.5 Securing DNS messages and zone transfer

Dynamic update messages need to be secured, especially when transferring DNS data over an unsecure network, such as the Internet. The same is true for the zone transfer protocols as well. A simple way to restrict, who can dynamically update or request a transfer of the zone file, is to use a filter to allow updates only from certain IP addresses. Due to IP spoofing, implementing security only by restricting IP addresses is not sufficient. Wellington (2000) describes how to secure DNS update messages, to allow only authorized sources to make changes to zone file. He proposes that TSIG (Transaction SIGnature) or SIG(0) could be used. The same protocols can be used to secure the zone transfer mechanisms.

4.5.1 TSIG and TKEY

TSIG, defined in RFC 2845 (Vixie et al. 2000), uses shared secret and one way hashing algorithms to provide transaction level security. It can be used in AXFR and IXFR transfers as well as to authenticate update messages received from approved clients or responses received from approved recursive name servers. TSIG introduces a new RR type, whose mnemonic is TSIG and type code 250. TSIG resource record is not an ordinary RR since it is dynamically computed for a particular transaction. Because of that it is called a meta-RR and should not be cached. Due to this, the TTL value of the TSIG record is set to zero.

TSIG operates as follows. Once the outgoing DNS message has been constructed, the message digest can be calculated. The calculated digest is then stored to TSIG RR, which is inserted to the additional data section of the DNS message. The header field ADCOUNT that specifies the RR count in Additional Data section is also updated to correspond to the new message length. If the messages size exceeds the maximum size that can be used with UDP when inserting TSIG information to the message, the message will be altered. The message is stripped so that only question and TSIG record is included. In the header, TC bit is set and RCODE is set to 0 and the response is sent to the requestor using UDP. When this message is received by the recipient, it should resend the original message using TCP instead of UDP. (Vixie et al. 2000, p. 5)

In a normal operation, when the recipient receives DNS messages with TSIG, it first checks that there is only one TSIG RR in the Additional Data section and that it is the last RR in the section. If these rules don't apply, recipient sends an error message to the sender. When the structure of the received message is correct, receiver copies TSIG RR from the additional data section to a secure location and removes it from the DNS message. It updates the message header as described in the previous paragraph. Recipient can now calculate keyed digest. For security reasons the received message must be discarded, if the algorithm name or key name is unknown to the recipient, or if the digest does not match. (Vixie et al. 2000, pp. 5 - 6)

Rules for security when using TSIG in DNS message transactions include the following. If server receives a message with TSIG RR, it is required to use TSIG also in the response. System time of the sender and recipient must be close enough, so that message is received in a time interval specified in TSIG RR. It is assumed that DNS servers are in correct time, so if BADTIME error is received from the client, server should not adjust its time based on the message. With the time check replay attacks are impossible. (Vixie et al. 2000, pp. 9 - 11)

Extra care should be taken to secure the shared secret keys. Hosts holding keys should be configured so that only users that need the keys for operations have access to them. Keys should be created using strong enough algorithms. Vixie et al. (2000) recommend that secret should be at least as long as the keyed message digest.

The challenge using TSIG is how to deliver shared secret securely to the other participant once it's created by another. RFC 2845 (Eastlake 2000b) defining the TSIG does not define any other key transportation mechanism than manual. Transaction Key (TKEY) RR defined by the RFC 2930 enables participants to securely establish shared secret keys without the need of manually transporting shared secret. The RFC defines multiple modes, but states that only the Diffie-Hellman mode must be supported by the DNS servers implementing TKEY. Other modes are optional.

4.5.2 SIG(0)

SIG(0) is an alternative protection method for dynamic update messages and zone transfer. In the name SIG(0), the SIG denotes signature and (0) denotes that the type covered field in the SIG resource record's RDATA portion is zero. Having the field zero tells that signature is transaction signature. If it is not zero, signature is used for RR authentication. The type covered field tells which types of RRs the SIG RR covers. (Eastlake 2000a, p. 2; Eastlake 1999, p. 18)

SIG(0) uses public key infrastructure (PKI). It is used as digital signature of the data transmitted like one can digitally sign an email and recipient can then verify the sender. Host possesses the private key and the public key is published in the DNS using KEY resource record (Eastlake 2000a, p. 3). It must be noted that in the server the private key used to sign messages is not the key used to sign zone information. The private key must also be available online to be able to sign update messages as they are sent, if it is not used to sign dynamic updates, it may be stored offline. (Eastlake 1999, p. 6) SIG(0) signs the whole outgoing DNS message, including its header. As TSIG SIG(0) also inserts its signature as the last RR to additional data section of the update message. (Eastlake 2000a, p. 5)

4.5.3 TSIG vs. SIG(0)

There are significant differences between TSIG and SIG(0). TSIG uses symmetric shared secret algorithm to create a message digest that is used to authenticate the message at the recipient. Digest is inserted into the additional data section of the update message. SIG(0) on the other hand uses asymmetric public key infrastructure to digitally sign the message. Signature is inserted to the additional data section as well. In TSIG, communication parties can be quite sure that other party understands the TSIG protocol, since secret keys need to be installed to both ends before it can be used. Using SIG(0) where public keys are stored into KEY RR, other party cannot be sure that the other one understands SIG(0) just from the fact that KEY RR exists. SIG(0) may also require an additional queries to be made to acquire the public key of the other party from some other DNS server. These reasons make SIG(0) computationally much more demanding compared to TSIG. SIG(0) should only be used when it is necessary to authenticate the requester or that it has privileges required. (Eastlake 2000a, p. 4)

Atkinson and Austein (2004) have made a threat analysis about DNS. In chapter regarding dynamic update they denote that, securing DNS update messages with TSIG does not scale very well into large ISP environments, since it requires configuring shared keys practically for every DNS client updating DNS information with dynamic update or zone transfer protocols. Using the same shared secret with multiple clients is not acceptable. It prevents the server from explicitly identifying each client and enables malicious clients to update some other clients' information if it learns or guesses the information needed.

4.5.4 Host security

Running DNS service on a host requires also a sufficient level of security from the host itself. DNS zone and configuration files, authentication keys and so on must be secured against unauthorized access coming from inside and outside. Basic rules of running services include that services should not be run as a root or an administrator user. File permission should be checked prior to starting the service, so that important files are not left with unnecessary read and write permission. Essential files have to have an up-to-date backup at all times in an offline location. Security of a host can be increased by pointing the root folder of the service to somewhere else than the root folder of the system. The technique is called the chrooting or sandboxing the service. (Aitchison 2011, p. 273)

Keys used in zone transfers and dynamic updates have to be kept in secure location. If PKI is used, the private part of the key pair has to kept offline when possible. There are cases like DNSSEC secured zone accepting dynamic updates, when the private part of zone signing key has to be online in order to sign the zone after updating its information. In cases like this the access to the keys has to be limited very strictly. The DNSSEC is described more in the following section.

4.6 DNSSEC

Upon receiving a reply to a DNS query message, the DNS server cannot be sure that the data received is correct. This is the case when using unsecure queries. The DNSSEC (DNS Security Extension) addresses the problem. Initially described by Eastlake (1999) and later updated and augmented by several RFCs, the DNSSEC specification is now implemented by the DNS root and increasing number of top level domains. Currently, in August 2013, 96 out of 316 top level domains are implementing the DNSSEC. (Aitchison 2011, p. 317; ICANN 2013)

DNSSEC is a set of techniques whereby a DNS resolver can verify authenticity and integrity of the query responses received from a domain implementing DNSSEC. To be able to benefit from DNSSEC, both the querying resolver and the zone queried must implement it, naturally. It uses PKI and special resource records to authenticate that data received can only have been sent by the requested zone, to check the integrity of the data and also to make sure that queried record really does not exist. Resource records mentioned here are the RRSIG (Resource Record Signatures), the DNSKEY and the

NSEC (Next Secure). It should be noted that DNSSEC cannot be used to protect DNS operations such as zone transfers and dynamic update. (Aitchison 2011, p. 318)

4.6.1 Securing a zone

To implement DNSSEC in a zone, the zone data has to be cryptographically signed with PKI. Using the private key RRsets are signed by calculating the digital signature from the content of the RRset. The signature is then stored to the RRSIG resource record, which is placed to zone file. The public key corresponding to the private key used to sign RRsets is published using the DNSKEY record, which is also added to the zone file. The third record related to DNSSEC, the NSEC, is used to chain all RRsets in a zone together. Querying a name from an unsecure domain that does not exist differs from a domain implementing DNSSEC. The authoritative name server of an unsecure domain would just respond with NXDOMAIN response code. This is not possible when DNSSEC is used, because DNSSEC signs RRsets, not response codes. NSEC RR is used to indicate to querier what is the next domain name existing in a zone file. Although domain names in a zone file may exist in random order prior to securing it, when DNSSEC is enabled, names are rearranged in to canonical order. NSEC record naturally contains the domain name of the RRset secured by NSEC, but also the next domain name in the zone file. This creates an obvious security threat. A malicious attacker can use NSEC to learn every RRset, thus domain name of a zone, by repeating queries for domain names that do not exist. NSEC3 addresses this problem. (Arends et al. 2005, p. 4,6; Vixie et al. 1997, p. 5; Albitz & Cricket 2006, p. 330-331)

NSEC3 record adds security against zone enumeration by hashing the domain names included in NSEC record. Hashing by itself does not create enough security, since domain prefix like www creates always the same hash, thus can be guessed quite easily. NSEC3 strengthens hashes by adding an optional salt to the name before hashing it. Another technique is to hash again already hashed name. Number of hash iterations performed to the domain name is included in NSEC3 record along with the hashing algorithm and salt. The salt used by NSEC3 for the zone in question is stored in NSEC3PARAM resource record. (Aitchison 2011, pp. 356 - 357; Laurie et al. 2008, p. 7)

Security does not come without cost. Upon receiving a query for to domain that does not exist, the authoritative DNSSEC server has to find the correct NSEC3 record for the reply, it has to add the salt to the name and perform hashing for defined number of iterations. This consumes a lot of server resources. Because of the resource consumption, NSEC3 enabled DNS servers are vulnerable to simple DoS (Denial of Service) attacks. Although salt and hashing is used, it is still possible to guess and calculate the original domain names. Thus it is recommended to use at least 64 bit random string as a salt. The salt should be changed occasionally to prevent pre-calculated dictionary attacks. (Aitchison 2011, p. 357; Laurie et al. 2008, pp. 30-31)

4.6.2 Updating zone information

DNS zone file needs to be re-signed every now and then. The most common reason is a change made to a resource record. Since these kinds of changes may occur very often, keys used to sign the zone have to be easily accessible. Although the most secure manner would be to store the zone signing keys in some offline storage, this is not possible in every case. If dynamic DNS update is used, the keys need to be online, to be able to sign the updated zone. Since the keys cannot be stored offline, it is recommended to generate long enough keys, so they cannot be hacked through calculation or guessing. Keys used with a zone may be divided into two to enhance security and to minimize changes to chain of trust. Zone signing key (ZSK) is used to sign resource record sets within a zone file. Key signing key (KSK) on the other hand is used to sign the root DNSKEY RR of a zone file. Re-signing the root DNSKEY RR affects to the chain of trust, since the signature of the zone changes. Using ZSKs and KSKs zone administrator can update resource records of a zone without the need to update the chain of trust every time. Updating the chain of trust requires always some effort from administrators of the parent zone. Using separate keys, KSK can be stored offline during normal operation, since there is no need to often re-sign the whole zone, while ZSK is kept online to be able to re-sign changed RRsets. (Kolkman et al. 2012, pp. 8-9)

4.6.3 Security islands and chains of trust

It cannot be required that every zone in the world implements DNSSEC. As stated earlier, the root and some of the top level domains implement it. Still there will always be unsecure zones in the DNS tree that create gaps to the chain of trusted zones. These gaps separate secured zones into so called islands of security. DNSSEC is designed to manage these gaps by using secure queries when querying from a secured zone and unsecure queries when querying from an unsecured zone.

In theory, to be able to authenticate the signed responses, validating resolver needs to have the public key of every secured zone. It would be practically impossible to get them safely and also to store a key for every secured zone. Trust anchors and chains of trust are used to resolve this. Every zone in the island of security, starting from the DNS root, can be authenticated using the public key of the DNS root zone. Similarly zones belonging to other islands, not reaching the DNS root zone, can be authenticated using the public key of the highest level zone of the island. The public key of the DNS root zone or the highest level zone of the island of security is called the trust anchor. Trust anchor acts as secure entry point (SEP) for the island of security and starts the chain of trust. Picture below depicts relations between SEPs and security islands. In the figure below, the public key of the zone example.com is the trust anchor of the island.

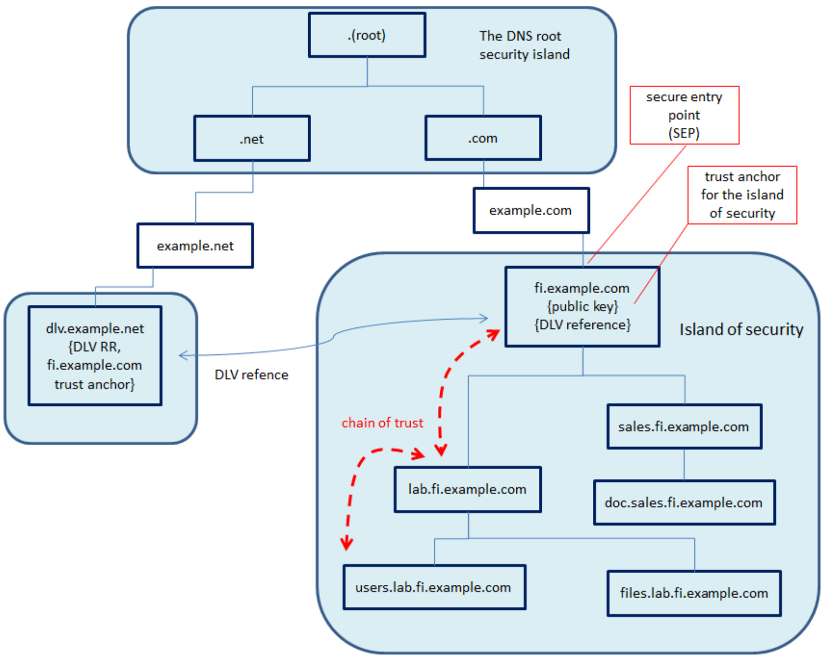


Figure 4.11. *Island of security, trust anchor, SEP and DLV*

Security islands are built by forming chains of trust using DS (Delegation Signer) resource records. In order to form a chain of trust between parent and child zone, both zones have to be signed first. Chain of trust is formed by adding child's DNSKEY record information to DS record in parent's zone file. Using the above figure as an example, the DNSKEY record information of the zone lab.fi.example.com is added to the DS record of the fi.example.com zone. (Aitchison 2011, p. 318-321; Arends et al. 2005, p. 15)

As in Figure 4.11 above, the chain of trust does not reach the DNS root zone. In order to maintain these islands of security, the public key or the trust anchor of the highest level zone has to be stored to the validating resolver, otherwise validating resolver cannot validate responses received from that island. As there are gaps in DNSSEC tree, there are and there will be numerous security islands. This means, that validating resolver has to get and store the trust anchor for every island, which burdens the server a lot. In order to overcome this challenge, the DNSSEC Lookaside Validation (DLV) has been implemented. Third party institutions, like ISC, maintain DLV domains to store trust anchors of the islands of security. Validating servers use these domains to verify the trust anchor, the public key of a zone. This way they do not need to store every trust anchor by themselves. Normally a validating resolver tries to validate the signature of a zone by looking for a trust anchor of the zone from among public keys stored to resolver. If not found, validating server tries to validate a zone with parent zone using the chain of trust. If neither method succeeds, as in the case of root zone of an island of security may be, validation has to be done against some other authority. DLV domains are implemented for situations like this. DLV domain uses DLV RRs to refer to a zone whose parent zone is not signed. DLV record format and function is identical to the DS record. The only difference is that when DS record refers to a delegated sub-zone, DLV record refers to a zone elsewhere in the DNS tree. Reference to a DLV domain is configured in the authoritative name server of a zone in question. When validating resolver learns about it, it sends DLV query to the authoritative name server of the DLV domain. If and when a DLV record is found for the zone queried the zone is considered secure and responses from it can be trusted. (Aitchison 2011, pp. 363-364; Weiler 2007, pp 1-2)

5 MULTICAST DNS

This chapter introduces multicast DNS in detail. Conventional DNS has to have servers configured to reply queries for the domain. As described in earlier chapters, multicast DNS does not need a server in order to function. Hosts running the multicast DNS service are clients and server to the service simultaneously. Using multicast DNS it is easy to quickly setup a network, which provides name service and maybe some other services on top of it. Apple has been using protocols similar to multicast DNS for decades, so it is not a surprise that the multicast DNS RFC 6762 that was released earlier this year, is written by employees of the Apple.

5.1 Multicast DNS in operation

Conventional DNS and multicast DNS are mainly alike. The main difference is the way the messages are sent and answered to. In conventional DNS messages are sent via unicast. Sender knows exactly the recipient of the message, since it is sent to a unicast IP address. In multicast DNS a message is sent via multicast. Message is received by every host listening to a specified multicast address on a local link. Another distinct difference is that in conventional DNS servers resolve the queries and answer to them. Due to a lack of the central authority in multicast DNS, every host connected to the same local link answers for itself.

Multicast DNS uses IPv4 multicast address of 224.0.0.251 as destination IP address when sending DNS queries for .local. namespace (IANA 2013a). Respectively, the IPv6 multicast DNS link local address is FF02::FB (IANA 2013b). Due to addresses mentioned are in local link multicast range, queries sent to the addresses are not forwarded outside the local link. Hosts can therefore be sure that any local link multicast message sent to or received from these addresses stay within local link.

In some situations multicast responders will use unicast when responding to a query. Such a situation occurs when a host is connected to network and has no prior knowledge about network resources. Host issues a query with unicast-response bit set telling other hosts that it accepts unicast responses to avoid large floods of unnecessary multicast responses. (Cheshire & Krochmal 2013a, pp. 11 – 12)

The specification also points out, that there may be some specialized applications when multicast responder uses unicast queries. Another not so special situation where unicast queries are used are legacy DNS clients that have been modified to co-operate with multicast DNS by sending local link messages to 224.0.0.251:5353 or [FF02::FB]:5353 and queries for other domains to unicast DNS server. (Cheshire & Krochmal 2013a, pp. 8, 13)

In any situation, when multicast responder receives a unicast query, it should somehow check that querier is within the same local link. Normally, if a query is determined being sent outside local link, it should be silently ignored. (Cheshire & Krochmal 2013a, p. 8)

5.2 Differences between the two versions of DNS

Although multicast DNS shares many features with conventional DNS, such as resource records and naming syntax, there are also many differences between the two. Cheshire & Krochmal (2013a, p. 51) present a list of differences, which is summarized in the following paragraphs. In addition to differences introduced earlier multicast DNS uses UDP/TCP port of 5353 whereas conventional DNS uses UDP/TCP port 53. Multicast DNS supports larger UDP datagrams, so it has to support TCP only because of legacy unicast communication. Since the namespace is shared, there is no SOA record to point out any single authority for it.

Next features of multicast DNS reduce the amount of multicast traffic needed. Multicast DNS allows more than one question in a query message. Multiple questions are placed into question section of a query message. The same query may be sent multiple times. Answers received to the preceding queries of the same message are placed into answer section of the subsequent query message. Thus a host, that sees its previous answers in a new query message, knows that it does not have to answer to the query again. If all known answers do not fit into first query message, querier sets TC bit on and places as many known answers to the subsequent message as possible. In case when known answers still do not fit into second query message either, the third and so on message is sent. Only the first query message contains question. Question section is empty in subsequent messages. (Cheshire & Krochmal 2013a, pp. 11, 23)

One significant difference is the lack of question in response message. Multicast DNS does not require that question section of the query is copied to the response, in fact, according to Cheshire & Krochmal (2013a, p. 14), the response must not contain the question and querier receiving a response containing the question should silently ignore the question section in the response.

5.3 Probing and announcing names

In this section the term name does not explicitly imply the name of a host. It may imply the name of a service a host is announcing as well. In case of two hosts probing for the same name simultaneously, the authority section of the query is used for tiebreaking. Hosts place all records and rdata to authority section, so in case of tiebreaking it can be used to select the winner. The winning host may announce the name by sending an unsolicited response. Newly registered resource records are placed in answer section. Host announcing new services must send at least two unsolicited message to make sure that everyone sees it. (Cheshire & Krochmal 2013a, pp. 27 – 30)

Conflicts are handled in cooperation with other participants since there is no central authority. If a malicious participant on the local link wants to interrupt the communication, it can do so. This can be prevented using authentication protocols like IPSEC or DNSSEC to accept messages only from trusted partners. (Cheshire & Krochmal 2013a, p. 52)

When a host needs to remove a resource record from network, it sends out a response message that contains the RR to be removed and sets the TTL value to zero. This message is called the good bye packet. Other hosts receiving such a message should not remove the record immediately, but to wait for one second, if some other host in network still announces the shared resource record. If a host is unable to send good bye packet before resource record is removed from network, the situation is the same as in conventional DNS. Querier sends out query messages for a while a stops trying, when no answers is received. In multicast DNS other hosts also see these messages and flush the record from their cache if no answer is seen. This is called Passive Observation Of Failures (POOF). (Cheshire & Krochmal 2013a, pp. 33 – 34, 38)

5.4 Extending over the link local boundary

Since IP networks are divided into several subnets for many reasons, multicast DNS should be able to extend over the local link boundaries. Bhandari et al. (2013) describe a multicast DNS gateway to overcome challenges of local link multicast scalability issues. Multicast DNS gateway may reside in a router. Its role would be to cache service advertisements and removals and respond to queries with the services in the cache. It has also the responsibility to forward unsolicited announcements from a local link to another to enable hosts in different local links to learn about resources in other local links. Gateway would be an active querier itself to be able to keep its cache fresh, but should use filtering to cache only services accepted by the policy.

Other implementations to announce services over local link boundaries are Hybrid Proxy by Cheshire (2013), which was described in section covering the DNS-SD and an experimental IETF draft written by Lynn & Sturek (2012), which modifies multicast DNS by using site local namespace and multicast addresses instead of link local.

6 HOME NETWORK NAME SERVICE

The majority of the IPv4 homenets are very simple. As discussed before, they consist of a single router that may include a built-in wireless access point. All devices at home are connected to the router by wire or wirelessly. The number of devices connected is relatively small and in most cases the only resource used in homenet, besides the Internet, is a fileserver that contains family photographs and other shared content. Due to simplicity of the network everyone can remember how to access the fileserver even when it has to be done by typing an IPv4 address.

From this point of view, there has been no need for the local name service in home networks. ISPs advertise DNS services to homenets for example via DHCP and router distributes the DNS information to the rest of the homenet devices. Should a homenet device need a public DNS name, dynamic DNS service can provide it.

As always, in the future, things will change. As discussed earlier in Chapter 3, the size of home networks will grow. New homes are filled with electrical household systems that control the heating, security and air conditioning and so on. Most of them are already network enabled and may be connected to homenet. Services provided by these systems can be accessed directly from the homenet. Since the fileserver is not the only device offering services to a homenet anymore, it becomes more challenging to remember how to access a certain service. To facilitate this problem the local name service needs to be introduced in the home network.

6.1 Requirements for the homenet name service

Chown et al. (2013) present basic requirements for homenet name service. Homenet name service has to support both lookup and discovery functions. A lookup uses direct query to a known service. For discovery two options are specified. It may use multicast messages, or a service where applications register and can be found. Homenet name service has to at least co-exist with the Internet name service. As Chown et al. (2013) state, there is a strong implication that homenet name service is DNS based or at least DNS compatible. Conventional unicast DNS and multicast DNS form a pair that could be configured to work in Zeroconf homenet environment. It could update an ISP DNS in order to enable global name service at home. DNS as homenet name service provides both resolving and authoritative name service. Authoritative name server may reside in home router, hosting the homenet zone and a secondary resolving name service maybe provided by an ISP or a cloud-based third party.

To protect against attacks some security measures should be applied. To be able to recover from a power outage or router change with as little damage as possible, the configuration and state information of a home router should be stored also to another location besides the router’s own memory. Since the information can be classified at least as confidential, the best location would be inside homenet. Cloud-service could be possible also, if it is considered secure enough.

6.2 Home network namespaces

If home users want to access their devices connected to the homenet from the Internet or anywhere outside homenet, a globally unique namespace is required. The namespace in question can be provided by the ISP or it can be a provider independent. Provider independent namespace services already exist and thus are not discussed any further here. ISP may offer a subdomain of some of its global domains for home customers when they are purchasing the connection. Home customer may have the option to select or even suggest a subdomain he wishes to have.

Global namespace may not be available for various reasons. In this situation homenet is required to choose and use a local namespace, significant only in local network. According to Chown et al. (2013, p. 33) homenet should not use .local. namespace, since it has a special meaning to certain link-local scope protocols, such as multicast DNS. The future homenet architecture should prepare for a multi-subnet environment. For these reasons homenet is required to choose a different domain. Two possible approaches to select a local namespace are described next.

The first one is an Ambiguous Local Qualified Domain Name (ALQDN) space, for example .sitelocal. or another appropriate name reserved for this purpose. Problem using this kind of namespace arises when device moves to another homenet with the same name. Names learned by the device in the actual homenet may not work in the other homenet although the namespace is the same. This may lead to misunderstandings and confusions. In practice, underlying service discovery protocols should be able to manage these situations without user intervention. (Chown et al. 2013, p. 33)

The second approach is to use a Unique Locally Qualified Domain Name (ULQDN) space. The syntax of it would be .<UniqueString>.sitelocal. The unique string could be generated in various manners. It could be adopted from /48 ULA prefix assigned to the homenet or it could be user defined. Device running the name service should generate at least a default value as last resort. In any case the unique string should remain unchanged even if the home router or a device running the name service goes through cold start. (Chown et al. 2013, p. 33)

In general, since it is likely that home users want to access homenet resources from outside homenet, homenet name service should follow the rules and conventions of the global name service. Homenet should support for several namespaces simultaneously. Segmentation of the namespace in use should be made possible through the support of hierarchical namespace management. User may not want that all homenet devices are visible to the Internet. A split view should be available to address these situations to be able to hide or not show a certain devices to the Internet. Reverse lookup service could be useful for logging purposes. With reverse lookup, device logs could be made more readable to user, when displaying the device name in the log instead of or parallel to IP address. (Chown et al. 2013, pp. 33-34)

6.3 Trust anchor for home network

Behringer et al. (2012) propose a trust anchor, like the one discussed in Subsection 4.6.3, for homenet in order to distinguish the borders of a homenet. They say that a homenet must be aware of its borders. In order to accomplish this, an autonomic device such as a router is selected to function as a registrar. Selection may be done manually or a device may select itself, if it discovers that there is no other autonomic device acting as registrar yet.

The registrar creates trust anchor for the homenet and grants domain certificates to other devices, thus acting as registration authority. Neighboring devices belonging to the same homenet are found using a neighbor discovery protocol, such as IPv6 neighbor discovery with some modifications. An autonomic device signs its neighbor discovery packets with domain, vendor or self-signed certificate. Since it already belongs to the homenet domain, it can invite its neighboring device to join the domain also. Join request sent by non-domain device is proxied to the registrar by the inviting domain device. If a device is accepted, it must perform a certificate enrollment process in order to acquire the domain certificate. (Behringer et al. 2012, pp. 2 – 3)

Using trust anchor and domain certificate, homenet devices can limit services announced by them to be seen only by domain devices. The approach does not prevent malicious devices impersonating friendly device to be invited to domain. Again, this does not differ from current security models in the homenet. (Behringer et al. 2012, p. 3)

6.4 Naming homenet devices

Nowadays ordinary people can remember a few IPv4 addresses if needed, like they can remember a few the most important phone numbers. IPv6 address is, on the other hand, much longer and for ordinary people it looks very random. As discussed earlier, the number of devices connected to home network will increase and migration to IPv6 will happen. These facts create an obvious need for naming devices connected to homenet instead of using only IPv6 addresses to address these devices.

Objective for IPv6 homenet is to be as user friendly as possible, meaning that user should not need to configure an IPv6 address or a host name to the device he or she just connected to homenet, but only connect it and start using it. Accessing other devices and services in the homenet should not mean that user has to remember or even type a full length IPv6 addresses either.

In enterprise environments there are standard practices to name hosts using DNS. In homenet normally there is not enough knowledge or it is not economically reasonable to setup such a heavy DNS service to provide names for hosts connected. Therefore other manners to organize the name service in homenet have to be discovered. The following sections present different possibilities to name devices in a home network.

6.4.1 Corresponding Auto Names for IPv6 Addresses

Kitamura & Ata (2012) describe a mechanism to obtain a name for every IPv6 address a host possesses. The mechanism is called Corresponding Auto Names for IPv6 Addresses. The idea is to distinguish every interface of a host by the last two characters of the interface MAC address.

Naming scheme

Format for a name corresponding to an IPv6 address is "<P><I>-<NGI>".

- <P> stands for type of prefix.
  - G is global, U is ULA and L is link-local address
  - Other characters may also be used if multiple prefixes for the same scope exist.
- <I> stands for Interface ID
  - 0 is for EUI64-based address
  - 1-9 is for manually set addresses
  - a-z is for automatically generated and set addresses except EUI64-based
- <NGI> stands for Node (Interface) Group ID
  - Shown as XYZ format
  - XY is the last octet, the last two charters of the interface MAC address
  - Z is a suffix character used to avoid collision if XY are the same on two interfaces of the same node. Default is z, if collided, the next used is y, x…

Mechanism is able to distinguish between automatically generated EUI64-based and manually entered addresses. The figure below depicts how names are generated from IPv6 addresses according the rules above.

```
Node A:         Literal Address                          Auto Name
                ----------------------------------       ---------
 MAC Address:   00:0d:5e:b8:80:7b
                ----------------------------------
 LL-Address:    fe80::20d:5eff:feb8:807b%fxp0         -> L0-7bz%fxp0
 ULA:           fd01:2345:6789::20d:5eff:feb8:807b    -> U0-7bz
                fd01:2345:6789::1234                  -> U1-7bz
 Global Addr:   2001:DB8::20d:5eff:feb8:807b          -> G0-7bz
                2001:DB8::1234                        -> G1-7bz
```

Figure 6.1. *Corresponding Auto Names (Kitamura et al. 2012, p. 5)*

If the scope of Auto names is wider than local link, the mapping tables of MAC address - <NGI> value and Prefix - <P> value are maintained to avoid collisions and manage mappings. (Kitamura et al. 2012, pp. 10-12)

Auto naming is not tied up to any specific name service. Depending on a scope of Auto Name, range from hosts file to Internet DNS can be used as name service. In general, the scope of Auto Names is local, not global. (Kitamura et al. 2012, p. 15)

Name generation and registration

To be able to generate and register names automatically, two mechanisms are needed. The first, called detector, detects appearance of new IPv6 addresses in the network and the second, called registrar, checks the appeared addresses, generates Auto Names and registers them to name service. (Kitamura et al. 2012, p. 16)

Detector uses DAD (Duplicate Address Detection) to detect newly appeared addresses (Kitamura et al. 2012, p. 16). DAD is a process used when generating and assigning a new IPv6 address to an interface. Prior to assigning a new address to an interface host must, according to RFC4862, send out a Neighbor Solicitation (NS) message. The message has the generated address as destination address. If the host receives an answer, it knows that the generated address is already in use and it must generate a new one and repeat the DAD process. (Thomson et al. 2007, pp. 13-17) Detector captures DAD messages and detects new IPv6 addresses in the network. Detected information is then sent to Registrar for further processing. (Kitamura et al. 2012, p. 16)

Registrar's responsibility is to check the address information received from the detector. It uses reverse resolving to check that the address received does not already have a name. With reverse resolving, the registrar queries name for the address. If an entry does not exist, registrar prepares a name for the address. Regular resolving is used for the prepared name to check for duplicate name entries. If a duplicate entry exists, registrar prepares a new name and performs duplicate check again. The process is repeated until a unique name is found. Once a name is found to be unique, registrar registers both regular and reverse resolving entries for the address and the name just found to be unique is registered to the name server. (Kitamura et al. 2012, pp. 16-17)

The proper placement of the detector and registrar ensures that Corresponding Auto Names mechanism is very flexible. Detector should be placed to every IPv6 subnet where hosts are connected. Registrar should be located parallel to name server to the same network segment. Corresponding Auto Names for IPv6 Addresses already have a working implementation described in the document. It uses BIND (Berkeley Internet Name Domain) and other common software. (Kitamura et al. 2012, pp. 17, 19)

6.4.2 Other naming schemes

When operating system is installed to a computer, it asks the user to name itself. It usually has a proposal for the name as well, so user does not necessarily have to change anything. Other appliances are pre-named as well. Printers, file servers and so on have a generic name given by the manufacturer.

Users may also want to be able to change the device name by themselves. This is enabled by default in operating systems. The user assigned device name may be the name that is updated to name service as well, or it may be a user friendly label distinct from the real device name.

Depending on the protocols used in a network, they may also provide a name for a host. Protocols such as multicast DNS generate and announce a name for a host and its service as discussed earlier.

6.5 Comparison between different name service methods

This section compares the name service methods that could be used in a homenet. First two are protocols that dynamically update name service information. The third is a manual method.

6.5.1 DNS and mDNS

Conventional DNS is used in current IPv4 home networks, although home users do not have to do anything to be able to use it since ISPs handle that part.
It could be used also in IPv6 home networks, but it has to be made as easy to use as possible. A web UI (user interface) could be set up, where a user can assign globally unique names to homenet devices. Conventional DNS is a protocol that every operating system running at the moment will understand. So using it would not mean any extra administrative work.

Multicast DNS does the previous step automatically using pre-defined values. At the moment every operating system does not have mDNS enabled by default. Some legacy operating systems that still exist in home networks do not have it installed or the mDNS software does not even exist for them. This is one reason multicast DNS may not be used in every homenet yet.

6.5.2 Hosts file

File named “hosts”, located in /etc directory in most operating systems, is the file used for operating system name resolution. In Unix based operating systems the file is the preferred source for name resolution for the host by default. Preference in Windows based systems varies depending on the version and configuration. In general, the order of the name resolution is local cache, DNS server, hosts file and Windows own methods, like WINS and LLMNR (Ts et al. 2003)

Administrator may add new IP address-name pair entries to the hosts file, which operating system then uses to resolve the IP address for the user entered mnemonic name. The file is local and meaningful only to the particular host.

In a small network with only a few hosts and infrequent changes, hosts file might be the most practical way to enable name service, for advanced users. Administrator only needs to add every host in the network to the file and copy it to every other device. For ordinary home users this method is not recommended.

6.6 Connecting homenet domain to the Internet DNS

Although users may want to be able to access some services residing at home from outside the homenet, some services should be accessible only within the homenet. Users accessing homenet services from outside favor the use of mnemonic names instead of IPv6 addresses. Considering previous requirements, there is a need for access control and name service between the homenet and the Internet. Next sections introduce ways to connect the homenet name service to the Internet name service or DNS.

6.6.1 IPv6 Home Network Naming Delegation Architecture

Cloetens et al. (2012) have written an IETF draft titled IPv6 Home Network Naming Delegation Architecture that describes an architecture to connect the homenet name service to the Internet name service. The key in this architecture is that homenet name service is hosted by the home router. It acts as an authoritative name server for the homenet zone and communicates with ISP’s DNS to keep its DNS up-to-date. In other words, name service requests from the Internet concerning homenet domain, are delegated by ISP’s DNS to the homenet router, which replies to them. The advantage of this architecture is that it hides the homenet topology and devices from the ISP. ISPs most likely do not want to make large investments to their DNS infrastructure only to be able to serve homenet customers. Using the delegation architecture ISPs can distribute the capacity needed for the DNS with customers.

There is, as usual, also disadvantages using the home router as an authoritative DNS server for the homenet. DNS traffic being redirected to home router by ISP’s DNS server burdens the subscriber line. This may lead to situation when DNS queries are lost due to an insufficient bandwidth. On the other hand, due to technical evolution, the bandwidth of the subscriber line has increased to a level where few DNS messages should not be a problem. Another disadvantage is that the architecture in question exposes home router to the Internet making it more vulnerable to different kind of attacks, for example denial of service attacks together with limited subscriber line bandwidth. (Cloetens et al. 2012. pp. 4-5) Basic DNS rules state that for a domain two separate name servers have to exist. So a home user has to either set up another host to serve as a secondary name server or purchase the service from a service provider. The next section introduces an alternative solution to connect homenet name service to the Internet without exposing the home router to the Internet.

6.6.2 Front End Naming Delegation

This section presents another method described by Migault et al. (2012). IETF draft is titled IPv6 Home Network Front End Naming Delegation. Here the home router manages DNS zone for the homenet and outsources queries coming from the Internet to public server residing in the Internet. It is designed to favor unmanaged operations and refers to the Home Networking Architecture for IPv6 by Chown et al. (2013) for naming recommendations. The draft presents following requirements:

- DNS queries for the homenet must be responded by the public DNS servers, when issued from outside the homenet
- The home router, by default, must not accept any DNS queries from outside the homenet
- IP address of the home router should not be publicly published
- DNS queries for the homenet must be responded by the home router, when issued from the homenet
- The home router must be able to update homenet zone hosted in the public DNS servers.
- The home router should be able to provide different views. At least one for the homenet nodes and one for the nodes outside the homenet. (Migault et al. 2012, p. 5.)

The idea is to have at least two different views for the home network. First one, called the homenet view, includes all devices residing in the homenet. The second, public view includes some portion of the devices in the homenet, which are to be publicly published. By public view homenet users may allow access to some homenet services from the outside, while preventing the rest. In some cases the homenet view and the public view may be the same, thus all homenet services are allowed to be visible in the Internet.

While the home router is the most logical place for the name service, it may also reside in a separate device. The document is written so, that the name service resides in the home router. Considering the requirements and assumptions above, the home router holds all three servers needed, resolving and authoritative server for the homenet and hidden master public server.

Name service inside homenet

Resolving server is an instance to which queries from the homenet should be sent. It functions as a forwarder. Resolving server should receive queries only from the homenet. If the query from the homenet concerns the homenet, resolving server forwards it to the authoritative server. Otherwise the query is forwarded to public DNS servers.

As the name homenet authoritative server suggests, the server hosts bindings between FQDN and IP address for the homenet. Unless cached all queries concerning the homenet should be handled by the authoritative server. Homenet authoritative server is responsible for the inside DNS zone and therefore for the homenet view.

Synchronizing homenet name service with Internet DNS

Hidden master public server is responsible for the public view. The document states, that the hidden master public server can update public view to public DNS servers in two ways. Using Dynamic DNS update may be used as well, but the document describes how the DNS slave-master architecture is used to synchronize public view to public DNS servers.

In the architecture the home router operates as DNS hidden master server and public DNS server as slave server for the homenet domain name. The master-slave configuration has to be agreed in advance with the homenet administrator and the provider of the public DNS servers. In the master-slave configuration, the home router hosts the public zone file and uses AXFR and IXFR transfer mechanisms and NOTIFY messages to update the public DNS server data. Upon receiving a NOTIFY message, a slave server can check if it needs to initiate a zone transfer. According to the draft, a home router must send NOTIFY messages only when necessary. A mechanism to enable this is to check the SOA record of the public DNS server first. If it is identical, no messages will be sent. The check needs to be performed, for example, when a router is rebooted. (Migault et al. 2012, pp. 9 – 10.)

Synchronizing data between a home router and a public DNS server must be secured, at least for integrity protection and authentication. The draft introduces a few protocols for this, for example TSIG and SIG(0), but recommends using IPsec and IKEv2. (Migault et al. 2012, p. 10.)

6.6.3 Updating homenet IP address to the name service

Usually ISPs do not provide static IP addresses to home customers. This is the case with current IPv4 implementations and most likely will be the case also with IPv6. This creates a challenge when using an Internet connected name service at home. Regardless of the name service implementation, home router must update its IP address to the DNS server every time it changes.

DNS update message can be used to update the resource record or records containing the IP address of the home router or a name server in home network. In practice, the home router sends an update message, where it first deletes the current resource record and then adds a new record containing the new IP address. Resource record type containing the IP address for a name is AAAA for IPv6 and A for IPv4.

Using non-secure DNS updates, this may be difficult, when an ISP uses an IP address based access control. Using secure, TSIG or SIG(0) based authentication an ISP does not have to depend on IP addresses, but to authenticate a customer through a shared secret or a public key.

7 HOME USER EXPERIENCE

This chapter concentrates mainly to customer or end user point of view, how he or she sees the service and what would be the easiest way for an ISP to implement a name service to home customers. ISP point of view is discussed shortly to be able to clarify issues concerning domain names, secure zone transfer and authentication of the home user. Different DNS topologies are compared at first. Then there is discussions how to name homenet devices and what type of domains can be offered to home customers. Last parts concentrate to ownership of the home router and mobility of the home network.

7.1 Comparison of DNS topologies

Here is a comparison between different DNS topologies. Every implementation includes a figure and a brief explanation of the pros and cons.

7.1.1 Name servers in the ISP network

The implementation Figure 7.1 represents is the same as the situation is nowadays in the IPv4 home networks. Both servers the master and the slave are located in an ISP network. This implementation has the major problem that the thesis is addressed to solve. Homenet name service should be available at all times, including times when the connection to the ISP is down. There are some workarounds, for example using multicast DNS in the homenet, but cooperation of mDNS and conventional DNS has no standard yet.

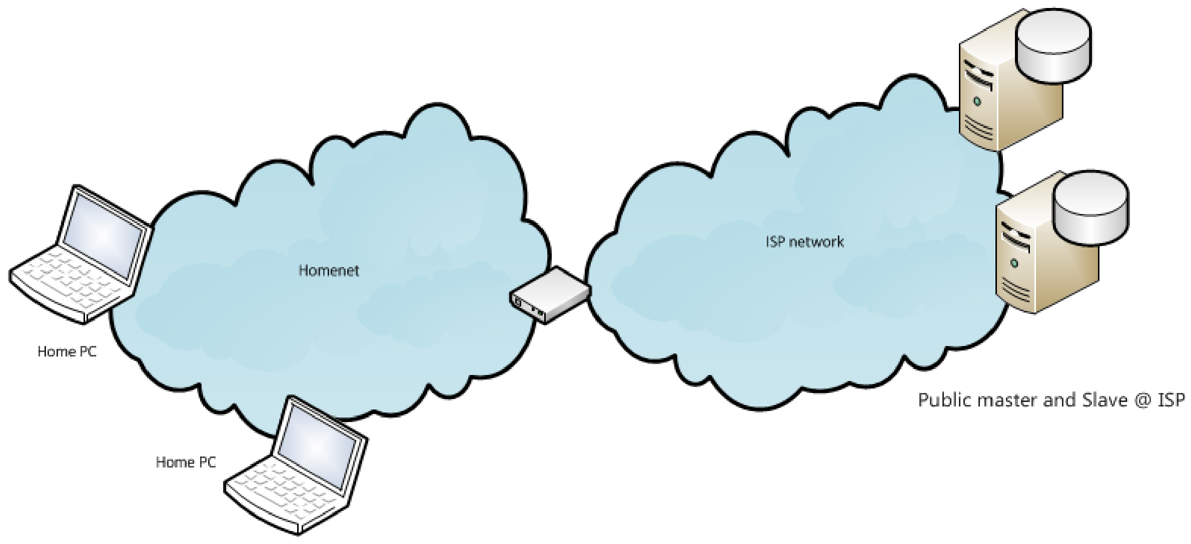


Figure 7.1. *Name servers in the ISP network*

Using this implementation there is the need for some ISP service portal through which customers are able to update the DNS information. Dynamic DNS updates may be used alongside to enable home devices update their IP address and name information for the domain. The good thing is that homenet is not revealed to the Internet, at least by the DNS.

7.1.2 Public master at homenet

In this implementation, as seen in the figure 7.2, the master name server for the homenet domain is located in the home router or in a separate device. The slave server is managed by the ISP. In this setup all queries regarding the home domain are forwarded to the name server at home according to the DNS architecture.

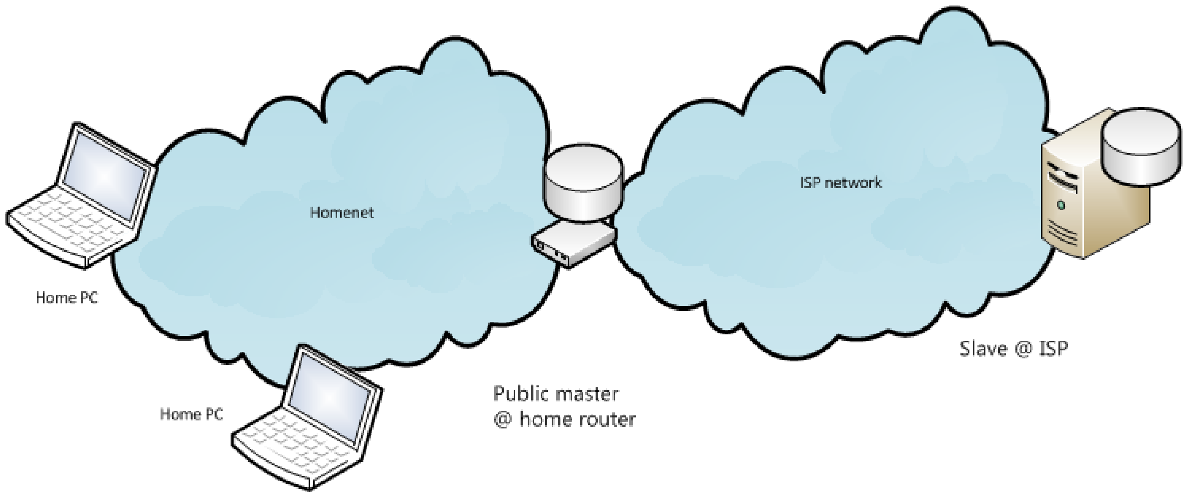


***Figure** 7.2. Public master at home, slave at ISP*

Pros for this implementation are that the name server is close to the main users of the domain, so the response time to queries inside the homenet is very low regardless of the bandwidth or utilization of the Internet connection. Outage in the Internet connection does not affect to name service within the homenet.

Having the public master server at home is also a disadvantage. Although DNS traffic for the home domain consumes only a fraction from the total bandwidth, it predisposes the Internet connection to denial of service attacks, especially when having lower speed connection. Another downside is that the host running the name service at home is exposed to the Internet. Malicious Internet users may try to hack it merely for fun. If the host is compromised, the attacker may be able to alter for example home surveillance system or heating with the information stored to the host.

The implementation also conflicts with the requirement presented by Migault et al. (2012, p. 5.) and the RFC 6092. The RFC recommends security capabilities to CPE devices such as home router and states that: “By DEFAULT, inbound DNS queries received on exterior interfaces MUST NOT be processed by any integrated DNS resolving server.” (Woodyatt 2011, p. 8) All in all, it is not reasonable to place public master server to home network, since there are better implementations.

7.1.3 Hidden master at homenet

The previous implementation had features that exposed homenet to the Internet. This implementation with hidden master remedies those features. It is quite the same as Front End Naming Delegation introduced in the previous chapter. Figure 7.3 below illustrates the implementation. Here the master server still resides in the home router, but it is hidden from the Internet using the DNS configuration not publishing it in the zone data, only slave servers know the location of the hidden master.

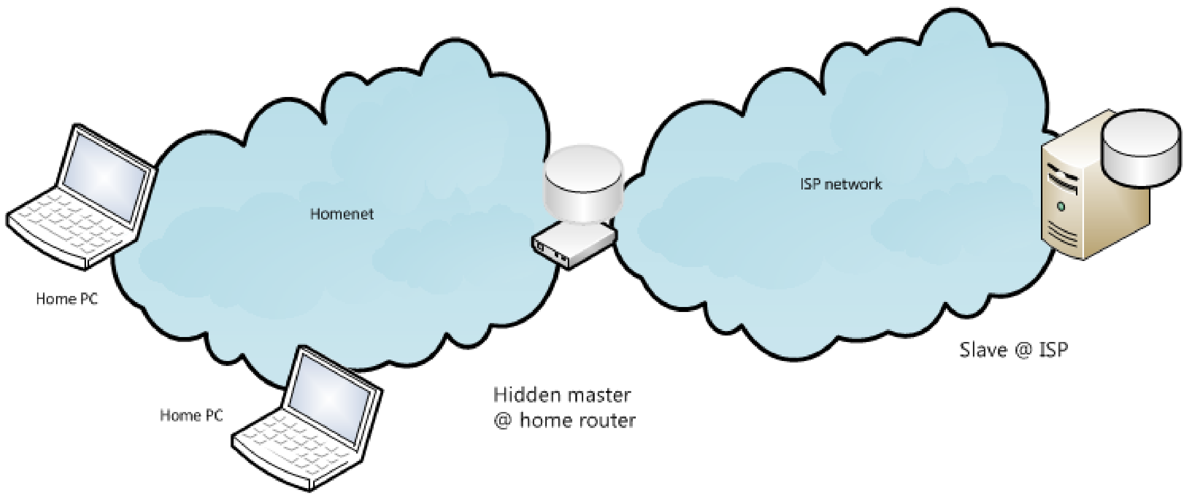


Figure 7.3. *Hidden master at home, slave at ISP*

This implementation prevents homenet to be vulnerable to denial of service attacks or hacking the home router by using the DNS information. The configuration has no affect to performance of the home router. On the contrary, it eases the load of the home router. With this configuration home customer may be provided a web UI for the home router to modify the name service at the home router, so an ISP service portal is not necessarily needed. After a while there may be no need for the web UI when name service protocols have evolved enough to cover all functions automatically.

Looking at the three implementations presented here the last one seems to have most benefits and lack any essential problems. For that reason it is the topology used in the lab network discussed in the next chapter.

7.2 Naming homenet devices

Naming home network devices should be automatic. Operating systems usually name the device it is running in or mechanism like Corresponding Auto Names, introduced in Subsection 6.4.1, may be used. Names given by operating system are rather descriptive to users, while naming scheme proposed by Corresponding Auto Names creates a name that does not describe the device at all. Chown et al. (2013) propose that device may have two different names, depending on with whom it is communicating with. For humans the names should be very descriptive, but for other devices it does not matter that much. This approach works if the name service is only local. When the local name service is connected to the Internet DNS and a user tries to contact a device at home from the Internet, almost certainly it does not work using the user friendly name alone, since the device user is using is in another domain. Domain is often hidden from users since it might only cause confusion. When visiting in another domain or a network, user should remember and use the full name of his homenet device, including the domain part.

A name given by operating system also has a down side. If in the future, for instance, every light switch may have an IPv6 address and a name, so there should be some rational manner to separate them. Otherwise one has a home with “light switch(1)”, “light switch(2)”, “light switch(3)” and so on, which is not very useful.

At the moment there seems to be no reasonable automatic naming scheme other than names given by operating systems. It is not a perfect one either, but is looks to be the best at the moment. Automatic names given by operating system usually can be modified through a user interface within the operating system. This makes them manageable, although not fully automatic.

Implementing an IPv6 home network, every device will have two separate IPv6 addresses. A device generates a link local address per interface from its MAC address and acquires a global scope IPv6 address by some means. If user wants to access the device from outside home networks, there is no choice, but to use global scope address. If name service is intended to be only local within home network, then link local addresses may be used as well. In relation to the topic of this thesis, the global scope address is to be registered to name service. There is no practical reason to register both addresses of a host, but nothing prevents it either.

7.3 Domains offered to customers

Advanced home customers may have their own domain already. Some may even have an own DNS server acting as master for the domain or zone. The required secondary DNS server is usually purchased from the ISP or some third party. Others may have purchased the whole package from a service provider. These domains are usually second level domains, such as example.com, where example. is the second level and .com. is the top level domain.

When a person or an organization registers a domain, it becomes responsible for it. This means, among other things that the person or the organization will have to take actions, if the domain is used to any illegal or inappropriate activities. These activities may consist of a hijacked host in the network that is sending junk mail or is used as a part of a bot network. User may also assign an inappropriate hostname to a host, which is then published in the Internet DNS. From the ISP point of view, it is legally much simpler if home customers register a second level domain. In that case ISP is not legally obligated in any way, though it provides the Internet connection. The authorities may request an ISP to shut down an Internet connection if it is used to break the law.

The other case is when an ISP delegates third or lower level domain from one of its own second level domains to home customers. Now the ISP is ultimately responsible for the second level domain and its delegates, so it has to take actions in situations described above. For these reason ISPs should carefully consider, what type of domains they accept and offer when planning to provide name service to home customers. ISPs should at least prepare to use filters to be able to remove or deny inappropriate names getting through to the Internet DNS. It is not an easy task, since DNS zone transfer protocols or dynamic update do not offer any moderation capabilities before publishing the names once a name has been inserted to the master zone file. One possibility is that ISP bundles the home router to the Internet connection subscription getting the control of the customer router at home. Customer may still have access to the router via a web user interface thus having ability to update name service information directly to the router's database. Another option is that ISP offers a self-configuration service portal for customers, where they can update their DNS information. If the home router is administratively managed by an ISP, it is possible to build filters for improper DNS information.

Looking at from the customer point of view a sub-domain delegated from an ISP domain is extremely binding. Customer can use the domain only when subscribing the connection from the ISP in question. When an ISP independent domain is used a customer can change ISP without the need of changing the domain. It just needs to update the necessary information in the zone data. This option is the least binding for the ISP as well, since it has no responsibility of the customer domain.

7.4 Ownership of the home router

Home router is owned and managed by the home user in most cases. There is a wide spectrum of hardware with different performance capabilities. If an ISP wants to build a reliable name service where DNS information transferred from homenet router to the ISP DNS servers can be authenticated, the ISP has to be sure that the server sending the information is the one it claims to be. If the home router is managed and owned by the customer, the ISP cannot be sure that server at home can be trusted. There is also an issue, how to reliably identify the customer acquiring the authentication for his home router in an service portal and how to make sure that the authentication key is not handed to or stolen by a third party. Issue discussed in the previous section, when building a filter for an inappropriate names, also favors the ISP owned home router.

A simple solution to these issues is that ISP rents the home router when customer is purchasing the Internet connection. ISP can now preconfigure the authentication keys to the router. Customer access to the router's web UI can be restricted so that authentication keys and other essential configuration are not revealed. If a customer changes the router, the name service no longer works, but the ISP is not affected. Currently available authentication algorithms also encourage ISPs to rent the home router.

7.5 Mobile homenet

Increasing numbers of home networks are changing their Internet access from fixed lines to mobile access. There are routers powerful enough to be used as DNS servers available, but operators offering mobile access favor small wireless broadband routers. They usually do not have enough resources to run DNS services other that DNS relay. These small hand held routers mostly lack the extensibility, so they cannot be modified to run DNS services.

The hardware issue may become significant for home customer that wants to enable name service over mobile network. Again, prices of the mobile broadband routers are dropping and they become more powerful containing more and more features. In a few years this might not be a problem anymore.

8 LAB NETWORK

Previous chapters introduced the background for homenet design and implementation. Based on those chapters the lab network was designed and implemented. It was designed to be scalable, although actually consisting of only a minimal number of devices. The chapter introduces the lab homenet topology, hardware and software used and configurations required to make it work. All relevant configuration files can be found in the appendices.

8.1 Topology and hardware

The physical topology shown below in the figure 8.1 consisted of two laptops (Home PC) acting as homenet user devices. Asus WL-500g Premium WLAN (wireless local area network) access point was used as a router between the homenet and the ISP network and a desktop PC acted as ISP DNS server. Linksys WRT54GL v1.1 WLAN AP was also tested, but it did not have adequate performance. In addition, a monitoring computer and a switch were used to view the messages sent between hosts. In some parts of the lab, when different software versions were tested, the monitoring PC was also used as a home user device.

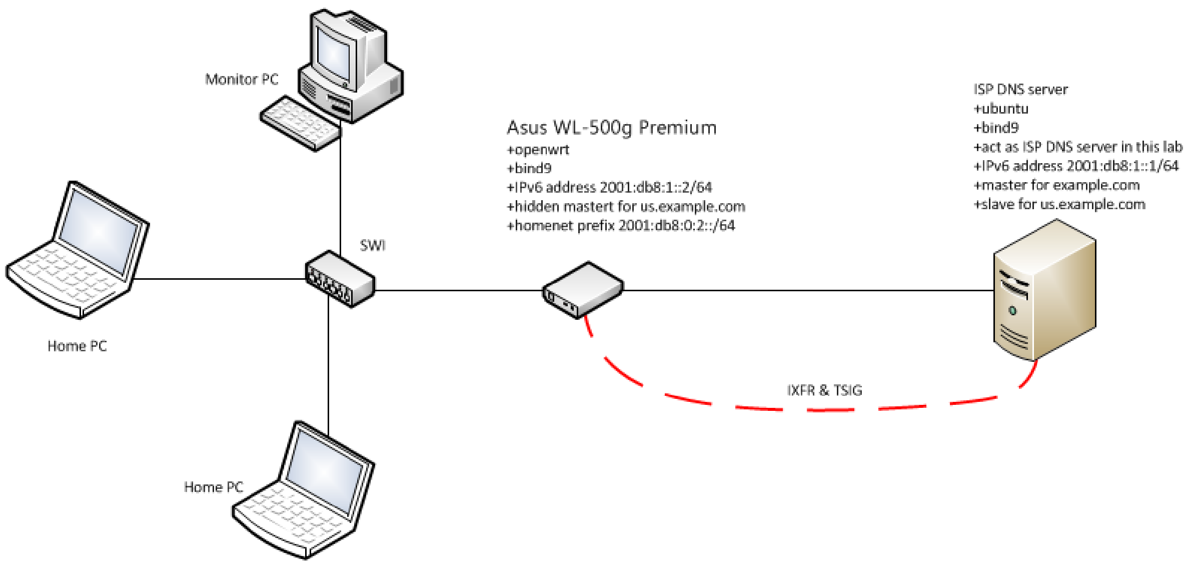

Monitor PC
Asus WL-500g Premium
+openwrt
+bind9
+IPv6 address 2001:db8:1::2/64
+hidden mastert for us.example.com
+homenet prefix 2001:db8:0:2::/64
ISP DNS server
+ubuntu
+bind9
+act as ISP DNS server in this lab
+IPv6 address 2001:db8:1::1/64
+master for example.com
+slave for us.example.com
SWI
Home PC
IXFR & TSIG
Home PC

Figure 8.1. *Lab network - physical topology*

In a real home network the switch between the router and end user devices is not normally required, since the router itself comprises a small switch. But larger wired homenet implementations require an external switch or additional router, since routers like the one used in the lab network comprise only minimal number of ports, usually four.

Logical topology of the lab network reflects a real live home network connected to an ISP network. The router between the homenet and an ISP network handles routing and access control. Access control means here firewall and remote access functions that limit access from the Internet. Figure of the logical topology is shown below.

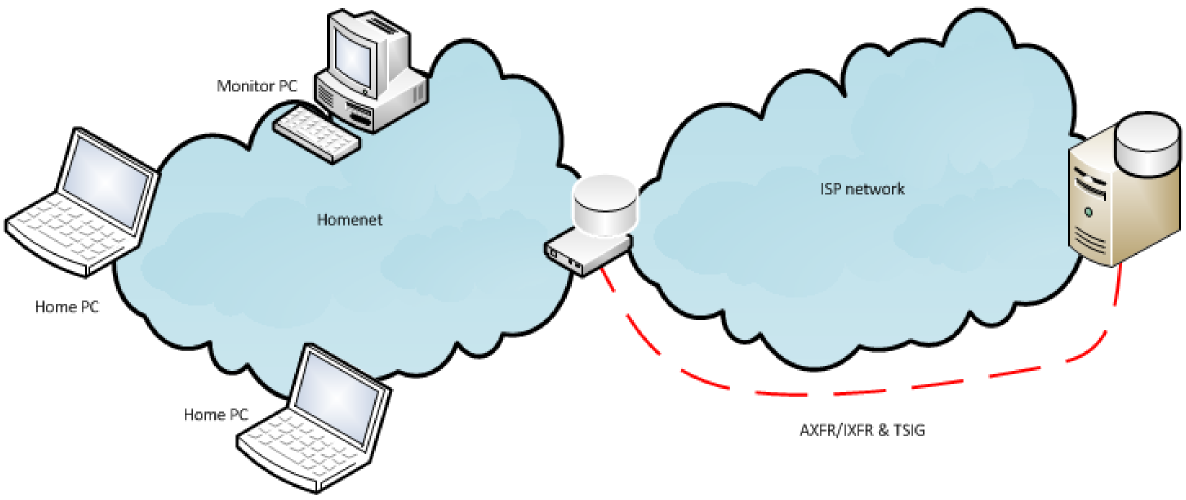


Figure 8.2. *Lab network - logical topology*

DNS topology chosen for the lab network was the hidden master - public slave. Homenet router acted as the hidden master for the home domain *us.example.com*, which was delegated from ISP's *example.com* domain. ISP's DNS server acted as the master and authoritative name server for the example.com domain and as a slave for the us.example.com domain. Zone data was designed to be transferred between the master and the slave using AXFR and IXFR zone transfer protocols and secured by TSIG authentication.

8.2 Software

Due to limited performance of the basic routers intended for home networks some software that can be run in a normal desktop or server hardware cannot be run in a homenet router. Most homenet routers include 4-8 MBs of flash memory and 16-64 MBs of RAM. This limits the size of the software it can run. In a desktop or server hardware this is not an issue, since they have multiple GBs of RAM and from hundreds of GBs to multiple TBs of hard disk capacity currently by default. To a desktop or server hardware the amount of memory may also be expanded easily. Software versions for all devices used in the lab homenet can be found in the Appendix 1.

8.2.1 ISP DNS server

The Ubuntu distribution of Linux was installed on desktop PC acting as the ISP's DNS server. On the top of Ubuntu the BIND software was installed for the DNS service. BIND is open source DNS software for the Internet. It has become the most widely used

DNS software in the world providing a stable platform scaling from a small office to the largest enterprise and ISP environments. (ISC 2013) No other mandatory software was needed in the ISP server.

Wireshark was installed to monitor packets in the interface facing the homenet. It was found very useful especially when debugging authentication issues between the master and the slave name server of the us.example.com domain.

8.2.2 Homenet router

Homenet router needed multiple software. The firmware of the access point had to be changed and various software had to be installed. Next section discusses the firmware and software used in the lab homenet.

Operating system

Due to limitations of the routers intended to home networks special small sized Linux distributions have been developed. There are many WRT individual distributions, but OpenWRT seems to be the most customizable and versatile one. It is a distribution that has been written from scratch. Nowadays there are over 3000 packages available to it. OpenWRT includes a web-based user interface as well as a command line interface. It is also free and seems to have the most extensive support over the device vendors. These were some of the facts, why the OpenWRT was selected as the operating system for the access point acting as the homenet router.

Advanced user have changed the original firmware supplied by the hardware vendor to one of the WRT distributions, because they have found those to be more stable, faster and configurable.

DNS and DHCP

Homenet router has to run at least DNS and DHCP services. For the DNS, Dnsmasq software was examined at first. Dnsmasq is a lightweight DNS forwarder and DHCP server designed for small networks (Kelley 2013). Quite soon it became clear that Dnsmasq could not be used, since it lacked essential features required in the lab network, such as TSIG support. Without the need of security features the Dnsmasq would have been used in the lab, because it comprises DHCP server and has some automated features for example to automatically create internal hostnames.

To get the required security features, the BIND was installed into the homenet router. Dnsmasq would have included the DHCPv6 server as well, but since it could not be used an alternative software had to be found. DHCPv6 servers available for OpenWRT were ISC's DHCP server, WIDE-DHCPv6 server and Dibbler DHCPv6 server. All three were explored. The ISC's and the WIDE server were noticed to be insufficient for the lab setup, so Dibbler DHPCv6 server was installed. OpenWRT firmware however included an old version of Dibbler, version 0.8.2, which contained a few bugs in essential parts for the thesis. The version was able to accept a fully qualified domain name from a DHCPv6 client, but despite the efforts a dynamic DNS update was not successful.

In order to build a tolerably functioning lab homenet the access point hardware had to be changed to a desktop PC running Ubuntu Linux. The main reason were software issues with Dibbler server mentioned above. After the change the latest Dibbler DHCPv6 server version was used and dynamic updates were successful.

Router advertisement

The original DHCPv6 specification defined by Droms et al. (2003) does not provide any routing information to DHCPv6 clients, hence other means are required to deliver the information. IPv6 standard solution is to enable router advertisement (RA). It can be configured to co-exist with DHCPv6 for example in a way that all other information is delivered using DHCPv6, but default gateway is delivered through router advertisement. RA was used in the lab homenet to ensure that clients running DHCPv6 software obeying the RFC were able to install IPv6 default gateway to their routing table.

8.2.3 Home user devices

Laptops in the homenet were running Ubuntu distribution of Linux and Windows XP. At first the DHCP client software provided by the operating system was examined and tested, but it did not have the abilities needed, so the Dibbler DHCPv6 client software was installed. Since clients were run in regular laptops, the newest version of Dibbler was available. But because no existing package was yet found, the software had to be compiled at first. Everything went well to the point when the client was supposed to run. But when the Dibbler client was told to start, a system error was displayed. The client package was recompiled and tested in two other computers with the same result. Hence an older software version had to be run. From then on laptops were using versions 0.7.3 and 0.8.2 of the Dibbler DHCPv6 client. To the laptop running Windows XP Dibbler DHCPv6 client version 0.8.4 was successfully installed. The configuration file was the same as used in Linux laptops.

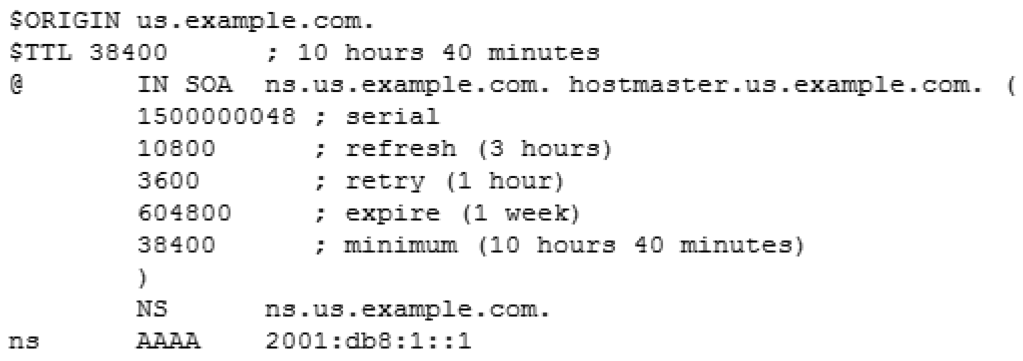
8.3 Configurations

This section describes configurations made to build a functioning lab homenet. Since DNS and DHCP are tied together very closely, they will be discussed in parallel. Besides DNS and DHCP configurations, also router advertisement is discussed here, since it was noticed to be needed to make homenet to work.

8.3.1 Zone configuration

BIND software provides default configuration for the default zones, such as local host and IPv4 reverse. However it does not provide any IPv6 configurations, so the IPv6 local host address had to be added to the default local host zone and the IPv6 reverse zone had to be created.

As described earlier, the domain us.example.com was configured as the homenet domain. Since the homenet router is acting as hidden master for its own zone, the zone must be configured so that its own IPv6 address is not included in zone data. Referring to the figure 8.1, the slave server was published in the SOA record and had its IPv6 address referred in the AAAA record as the authoritative name server for the domain us.example.com. The IPv6 address of the home router, and the hidden master for the us.example.com, is 2001:db8:1::2. The figure below represents the contents of the fresh us.example.com zone file.



```
$ORIGIN us.example.com.
$TTL 38400      ; 10 hours 40 minutes
@       IN SOA  ns.us.example.com. hostmaster.us.example.com. (
        1500000048 ; serial
        10800      ; refresh (3 hours)
        3600       ; retry (1 hour)
        604800     ; expire (1 week)
        38400      ; minimum (10 hours 40 minutes)
        )
        NS      ns.us.example.com.
ns      AAAA    2001:db8:1::1
```

Figure 8.3. *Zone configuration for the us.example.com*

The ISP server is the authoritative server for the domain example.com having the delegation for the homenet domain us.example.com. The ISP name server is also authoritative for the IPv6 reverse zone for prefix 2001:db8:1::/64. Since the ISP server is acting as the slave server for the us.example.com and 2001:db8:0:2::/64 zones, the slave configuration had to be added to the ISP server as well. The actual zone data for the us.example.com zone and the IPv6 reverse zone assigned to the homenet is transferred from the home router. Every time a change occurred in the homenet, a notify message was sent to the slave server to inform it that the data has changed. Notify mechanism was enabled for quicker convergence time. Otherwise zone data would have been updated until the time to live timer would have expired. Relevant DNS configuration can be found in the Appendix 2.

8.3.2 Home router

According to the author, the latest release 1.0.0RC1 of the Dibbler implements all features that RFC defining the DHCPv6 describes (Mrugalski 2013, p. 6). The newly released version contains a comprehensive user guide including complete example files, which can be copied straight from the document to the software configuration file. So the needed features were easy to enable by copying them from the user guide to corresponding configuration files. By default, an example file contained configuration for example for DNS, NTP (Network Time Protocol) and host naming.

Dibbler DHCPv6 server already includes an IETF draft phase extension to DHCPv6 for delivering routing information to clients. Dec et al. (2012) describes an extension to deliver routing information to clients using DHCPv6. The extension is implemented to the Dibbler DHCPv6 server since release 0.8.1RC1, although it uses yet unconfirmed option request option (ORO) to request and to deliver routing information to clients. (Mrugalski 2013, p. 20 - 21) As discussed in the software Section 8.2 the router advertisement feature was also enabled due to previous standardization reasons to enable current DHCPv6 client to obtain IPv6 default gateway. DHCPv6 and router advertisement configurations of the home router can be found from the Appendix 3.

Device naming

The Dibbler DHCPv6 server was configured to accept the hint from a client for the fully qualified host name, but only the host name portion was used by the server and the rest of the domain name was given in the server configuration and thus appended to complete the domain name. Using this method client will always get the correct domain suffix, including clients visiting in the homenet. Otherwise visitors may keep their home domain suffix and would not be able to use visiting network's resources.

Other options are also available in the Dibbler server for the name handling. A list of acceptable host names can be configured to the server. Upon receiving a hint for the host name from a client, server can perform one of the following actions. If the hint is not on the list of acceptable names, it will be silently rejected or a server can send an acceptable name to the client. Server can be configured accept any fully qualified host name the client suggests or accept only the host name portion, as in the lab setup. The last option is that server ignores the hint and generates a name for the client based on its address and appends the domain name given in the server configuration. (Mrugalski 2013, p. 67)

Dynamic updates

Home router hosting the us.example.com and IPv6 reverse zone for the homenet prefix were configured to accept dynamic update messages. In the beginning of the homenet lab it was unclear, which device would update host information to the homenet zone. There were two choices, the client connecting to the homenet or the home router. During the lab it became obvious that the only reasonable instance to perform dynamic updates for the both, forward and reverse zones is the DHCPv6 server running in the homenet router.

Using the configuration where the DHCPv6 server dynamically updates both forward and reverse zones protects the homenet name service much better than if hosts themselves could update their information. So in the DNS server configuration only the home router was allowed to update the DNS information.

In the Dibbler server version used first with Asus wireless access point, this setup did not work. When trying to enable such a configuration, the server did not start at all. After the hardware was changed, also dynamic updates were successful.

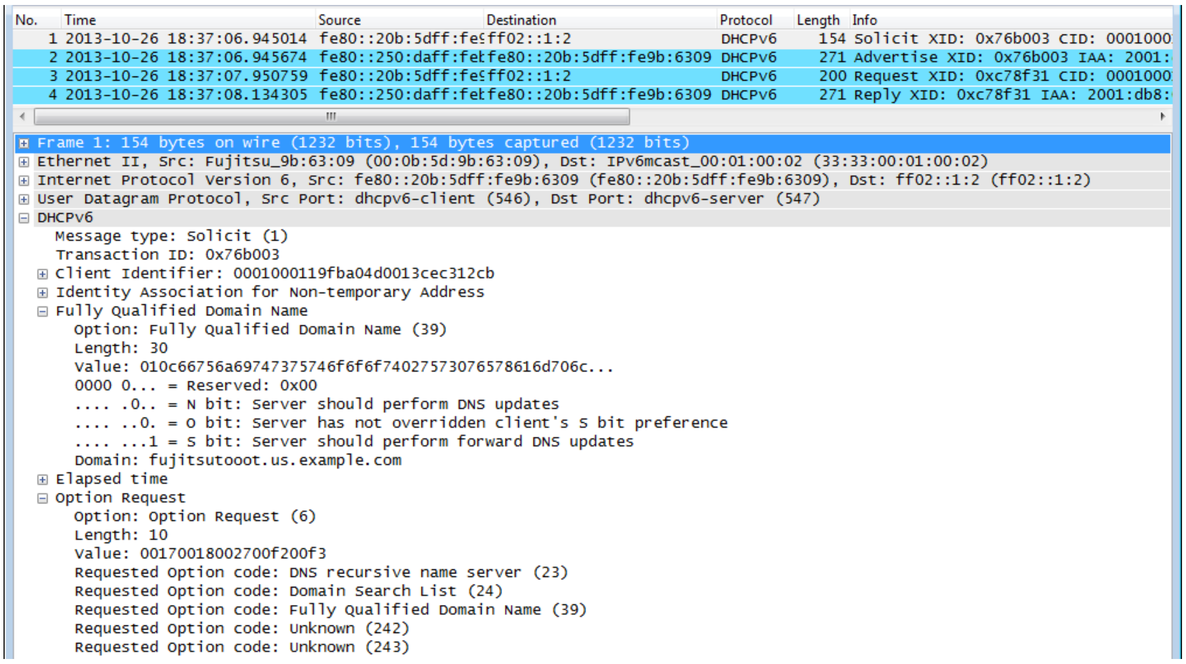
8.3.3 Home user devices

The whole purpose of the thesis was to research for methods to enable as automatic homenet as possible. By default a laptop or a tablet does not send its name to the DHCP server when asking for an IP address nor does the server request it. In the home network, the DHCP server does not usually even have a domain name available to append to the fully qualified host name as a suffix.

Naming of home user devices or laptops was done manually in this lab. Dibbler client does not include means to retrieve the host name used by the operating system. This might be possible by using an external script. A fully qualified host name was inserted into the client configuration file that can be found in the Appendix 4. The name was updated to the DNS by means described previously in the home router configuration section.

As discussed earlier, the RFC defining DHCPv6 does not contain conveying the routing information to clients, although it has been implemented to Dibbler server. Among other things this was a reason why Dibbler DHCPv6 client was used in laptops.

On Ubuntu laptop running Dibbler client 0.7.3, the software could not understand the routing information ORO, since it was not introduced until in version 0.8.1RC1. The other Ubuntu laptop using Dibbler client version 0.8.2 did accept the configuration and sent the routing information option request option number 242, as seen in the figure below.



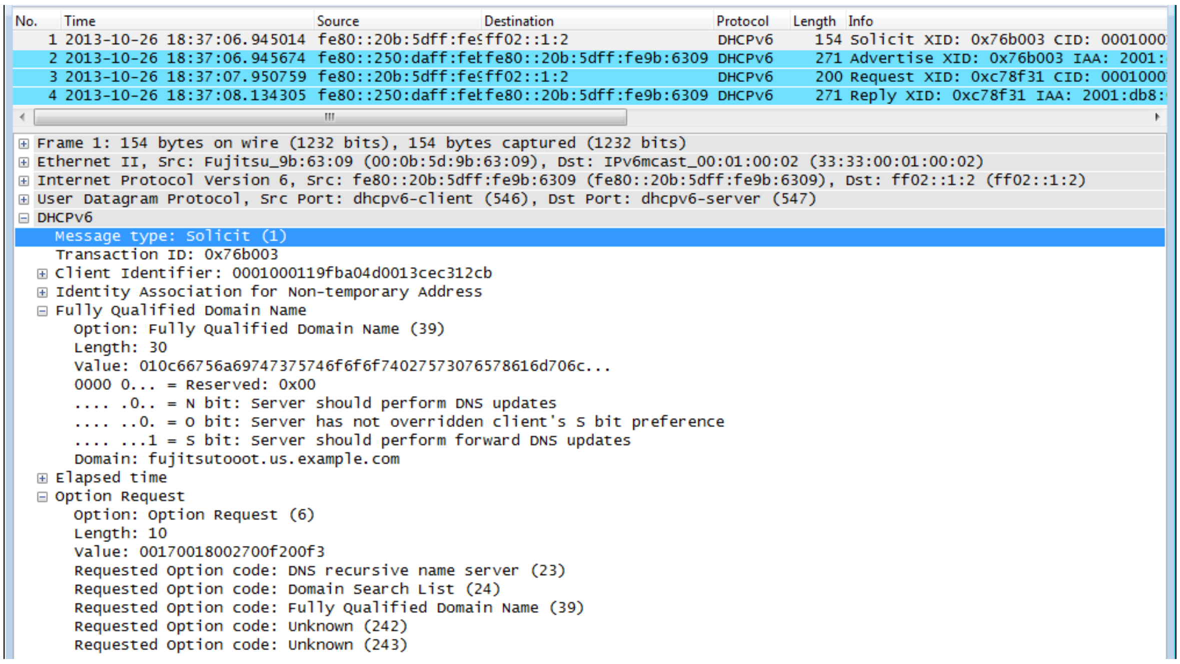
```
No.  Time                        Source               Destination              Protocol  Length Info
   1 2013-10-26 18:37:06.945014 fe80::20b:5dff:fe9ff02::1:2                    DHCPv6       154 Solicit XID: 0x76b003 CID: 0001000
   2 2013-10-26 18:37:06.945674 fe80::250:daff:fetfe80::20b:5dff:fe9b:6309     DHCPv6       271 Advertise XID: 0x76b003 IAA: 2001:
   3 2013-10-26 18:37:07.950759 fe80::20b:5dff:fe9ff02::1:2                    DHCPv6       200 Request XID: 0xc78f31 CID: 0001000
   4 2013-10-26 18:37:08.134305 fe80::250:daff:fetfe80::20b:5dff:fe9b:6309     DHCPv6       271 Reply XID: 0xc78f31 IAA: 2001:db8:

⊞ Frame 1: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits)
⊞ Ethernet II, Src: Fujitsu_9b:63:09 (00:0b:5d:9b:63:09), Dst: IPv6mcast_00:01:00:02 (33:33:00:01:00:02)
⊞ Internet Protocol Version 6, Src: fe80::20b:5dff:fe9b:6309 (fe80::20b:5dff:fe9b:6309), Dst: ff02::1:2 (ff02::1:2)
⊞ User Datagram Protocol, Src Port: dhcpv6-client (546), Dst Port: dhcpv6-server (547)
⊟ DHCPv6
    Message type: Solicit (1)
    Transaction ID: 0x76b003
  ⊞ Client Identifier: 0001000119fba04d0013cec312cb
  ⊞ Identity Association for Non-temporary Address
  ⊟ Fully Qualified Domain Name
      Option: Fully Qualified Domain Name (39)
      Length: 30
      Value: 010c66756a69747375746f6f6f74027573076578616d706c...
      0000 0... = Reserved: 0x00
      .... .0.. = N bit: Server should perform DNS updates
      .... ..0. = O bit: Server has not overridden client's S bit preference
      .... ...1 = S bit: Server should perform forward DNS updates
      Domain: fujitsutooot.us.example.com
  ⊞ Elapsed time
  ⊟ Option Request
      Option: Option Request (6)
      Length: 10
      Value: 00170018002700f200f3
      Requested Option code: DNS recursive name server (23)
      Requested Option code: Domain Search List (24)
      Requested Option code: Fully Qualified Domain Name (39)
      Requested Option code: Unknown (242)
      Requested Option code: Unknown (243)
```

Figure 8.4. *Routing information option request option*

Dibbler server responded to the option request option 242 by sending the default gateway's IPv6 address back to the client. Upon receiving the information, client had to execute an external script in order to install the default gateway information to the IPv6 routing table. The script is invoked in the event of insertion, update and removal of routing information from the IPv6 routing table when using Dibbler DHCPv6 client software. After the script was run the default gateway could be seen in the IPv6 routing table. The configuration file for a Dibbler DHCPv6 client can be found in the Appendix 4. The host name information is naturally a variable that needs to be changed per host and for older clients the routing information option request option has to be disabled due to incompatibility.

8.3.4 DNS update process

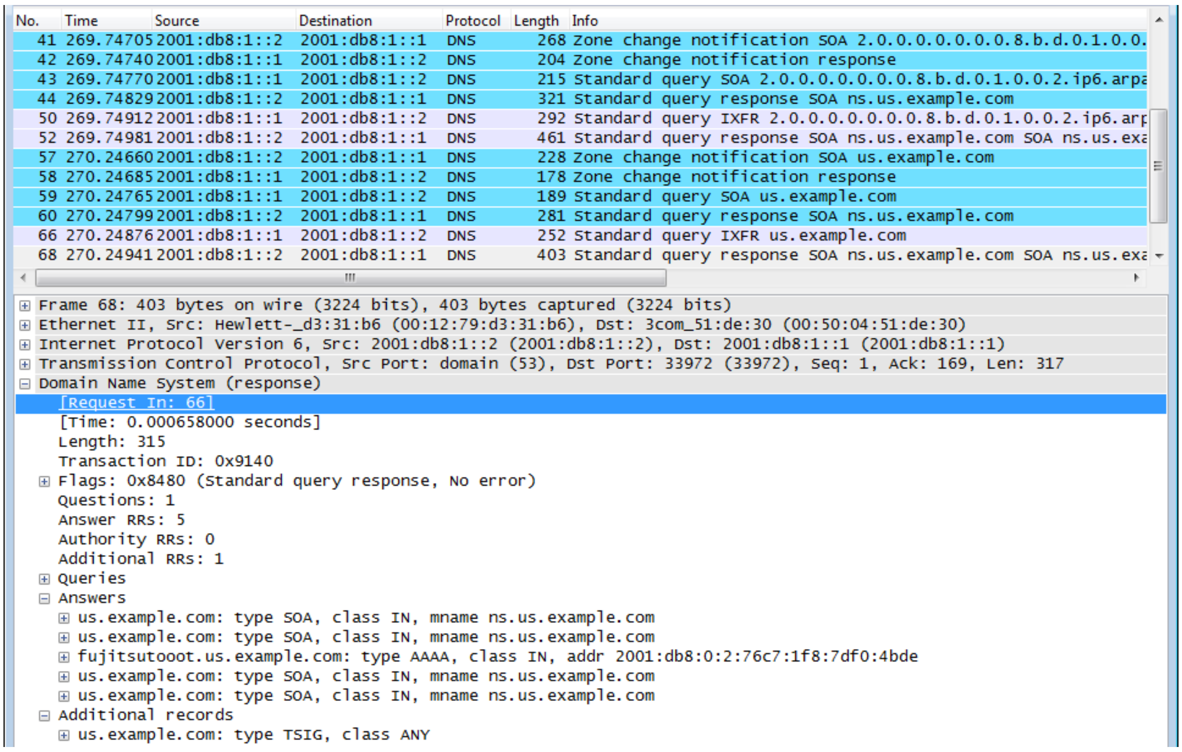
In this lab homenet the DNS update process begins when a host is connected to the homenet. Since the host name is delivered using DHCPv6, it will participate in the DNS update process. When a client has just connected to the network, it sends a DHCPv6 solicit message indicating among other things the host name it wishes to use. In the solicit message client also requests that server performs dynamic updates to both forward and reverse zones, the N and S bits. Servers respond with an advertise message to advertise their existence. The client selects one of the servers and sends a request message to it requesting the IPv6 address the selected server proposed. The same option request options and domain name request are also included that were sent in the solicit message. The last message is a reply from the selected server confirming the reservation of the IPv6 address and delivering other information, such as the DNS server. The figure below represents previously described DHCPv6 four way handshake. In messages 2 and 4 the server agrees to perform dynamic updates. The client is now ready to communicate with other hosts and the Internet.



```
No.  Time                        Source              Destination                Protocol  Length Info
   1 2013-10-26 18:37:06.945014  fe80::20b:5dff:fe9ff02::1:2                    DHCPv6       154 Solicit XID: 0x76b003 CID: 0001000
   2 2013-10-26 18:37:06.945674  fe80::250:daff:febfe80::20b:5dff:fe9b:6309     DHCPv6       271 Advertise XID: 0x76b003 IAA: 2001:
   3 2013-10-26 18:37:07.950759  fe80::20b:5dff:fe9ff02::1:2                    DHCPv6       200 Request XID: 0xc78f31 CID: 0001000
   4 2013-10-26 18:37:08.134305  fe80::250:daff:febfe80::20b:5dff:fe9b:6309     DHCPv6       271 Reply XID: 0xc78f31 IAA: 2001:db8:

⊞ Frame 1: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits)
⊞ Ethernet II, Src: Fujitsu_9b:63:09 (00:0b:5d:9b:63:09), Dst: IPv6mcast_00:01:00:02 (33:33:00:01:00:02)
⊞ Internet Protocol Version 6, Src: fe80::20b:5dff:fe9b:6309 (fe80::20b:5dff:fe9b:6309), Dst: ff02::1:2 (ff02::1:2)
⊞ User Datagram Protocol, Src Port: dhcpv6-client (546), Dst Port: dhcpv6-server (547)
⊟ DHCPv6
    Message type: Solicit (1)
    Transaction ID: 0x76b003
  ⊞ Client Identifier: 0001000119fba04d0013cec312cb
  ⊞ Identity Association for Non-temporary Address
  ⊟ Fully Qualified Domain Name
      Option: Fully Qualified Domain Name (39)
      Length: 30
      Value: 010c66756a69747375746f6f6f7402757307657861d706c...
      0000 0... = Reserved: 0x00
      .... .0.. = N bit: Server should perform DNS updates
      .... ..0. = O bit: Server has not overridden client's S bit preference
      .... ...1 = S bit: Server should perform forward DNS updates
      Domain: fujitsutooot.us.example.com
  ⊞ Elapsed time
  ⊟ Option Request
      Option: Option Request (6)
      Length: 10
      Value: 00170018002700f200f3
      Requested Option code: DNS recursive name server (23)
      Requested Option code: Domain Search List (24)
      Requested Option code: Fully Qualified Domain Name (39)
      Requested Option code: Unknown (242)
      Requested Option code: Unknown (243)
```

Figure 8.5. *DHCPv6 four way handshake*

The remaining part of the process concerns the home router and ISP name server. Now DHCPv6 server knows the fully qualified host name and IPv6 address of the client and sends a dynamic DNS update message to the master name server of the us.example.com domain. The master server updates its zone files and sends out a zone notify message to the slave server residing in the ISP. The slave server then initiates an IXFR zone transfer to update its zone data, respectively. There is no packet capture available from the dynamic update since it takes place inside the homenet router between Dibbler DHCPv6 server and BIND DNS server, but it can be observed from the servers' logs. Previously described process of zone transfer between master and slave servers is depicted in the following figure.



```
No.  Time       Source          Destination     Protocol Length Info
  41 269.74705 2001:db8:1::2   2001:db8:1::1   DNS         268 Zone change notification SOA 2.0.0.0.0.0.0.0.8.b.d.0.1.0.0.
  42 269.74740 2001:db8:1::1   2001:db8:1::2   DNS         204 Zone change notification response
  43 269.74770 2001:db8:1::1   2001:db8:1::2   DNS         215 Standard query SOA 2.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa
  44 269.74829 2001:db8:1::2   2001:db8:1::1   DNS         321 Standard query response SOA ns.us.example.com
  50 269.74912 2001:db8:1::1   2001:db8:1::2   DNS         292 Standard query IXFR 2.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arp
  52 269.74981 2001:db8:1::2   2001:db8:1::1   DNS         461 Standard query response SOA ns.us.example.com SOA ns.us.exa
  57 270.24660 2001:db8:1::2   2001:db8:1::1   DNS         228 Zone change notification SOA us.example.com
  58 270.24685 2001:db8:1::1   2001:db8:1::2   DNS         178 Zone change notification response
  59 270.24765 2001:db8:1::1   2001:db8:1::2   DNS         189 Standard query SOA us.example.com
  60 270.24799 2001:db8:1::2   2001:db8:1::1   DNS         281 Standard query response SOA ns.us.example.com
  66 270.24876 2001:db8:1::1   2001:db8:1::2   DNS         252 Standard query IXFR us.example.com
  68 270.24941 2001:db8:1::2   2001:db8:1::1   DNS         403 Standard query response SOA ns.us.example.com SOA ns.us.exa

⊞ Frame 68: 403 bytes on wire (3224 bits), 403 bytes captured (3224 bits)
⊞ Ethernet II, Src: Hewlett-_d3:31:b6 (00:12:79:d3:31:b6), Dst: 3com_51:de:30 (00:50:04:51:de:30)
⊞ Internet Protocol Version 6, Src: 2001:db8:1::2 (2001:db8:1::2), Dst: 2001:db8:1::1 (2001:db8:1::1)
⊞ Transmission Control Protocol, Src Port: domain (53), Dst Port: 33972 (33972), Seq: 1, Ack: 169, Len: 317
⊟ Domain Name System (response)
    [Request In: 66]
    [Time: 0.000658000 seconds]
    Length: 315
    Transaction ID: 0x9140
  ⊞ Flags: 0x8480 (Standard query response, No error)
    Questions: 1
    Answer RRs: 5
    Authority RRs: 0
    Additional RRs: 1
  ⊞ Queries
  ⊟ Answers
    ⊞ us.example.com: type SOA, class IN, mname ns.us.example.com
    ⊞ us.example.com: type SOA, class IN, mname ns.us.example.com
    ⊞ fujitsutooot.us.example.com: type AAAA, class IN, addr 2001:db8:0:2:76c7:1f8:7df0:4bde
    ⊞ us.example.com: type SOA, class IN, mname ns.us.example.com
    ⊞ us.example.com: type SOA, class IN, mname ns.us.example.com
  ⊟ Additional records
    ⊞ us.example.com: type TSIG, class ANY
```

Figure 8.6. *Notify and zone transfer messages*

Looking at the figure, at first, the homenet master name server, having the IPv6 address 2001:db8:1::2, sends a notify message for the reverse zone and the slave name server acknowledges it. Then the slave server starts the update process by querying for the reverse zone. Upon receiving the reply the slave checks, which information it needs to update and asks for it by initiating IXFR, the incremental zone transfer. The same process is repeated for the forward zone us.example.com. All messages are secured using TSIG as seen on the last row of the figure above.

8.3.5 mDNS

Using multicast DNS the homenet devices can find and advertise their services to each other. Multicast DNS is installed and configured by default for example in the current Ubuntu Desktop versions. Avahi-daemon is software that can be installed to enable multicast DNS in hosts that do not have it by default. Windows operating systems do not include mDNS by default, but there are a few software that can be used to enable it.

Multicast DNS was supposed to be examined also in the lab homenet, but due to delays in the schedule the mDNS could not be covered in the thesis. Yet it will be tested and demonstrated later.

9 RESULTS AND DISCUSSION

In this chapter the results of the thesis are introduced and discussed. Section 9.1 introduces the results divided into topics paraphrasing the preceding chapters. Section 9.2 contains the discussion about the results and the whole process of the thesis.

9.1 Results

This section represents the results of the theory and implementation part of the thesis. First two sections concentrate purely to the results of the theory part. The last two sections move from the theory to the implementation.

9.1.1 Device naming

First of all, devices in a home network should be able to associate a name to themselves. For device naming, there is an old method still widely used, where an operating system names the device when it is installed. Operating system usually asks if user wants to change the name generated by it.

Another method is to use a protocol such as multicast DNS to name the device. The protocol relies on naming methods used in zero configuration networking. The name given to a device may be the same as the one the operating system suggested during the installation. A significant difference is however the ability of multicast DNS to change the name automatically in co-operation with other devices in the network if two devices are using the same name.

Then there are some new visions about naming devices based on their physical address or addresses. This is the most complex method found and does not give user a chance to modify the automatically generated name. It may be processed further to use automatically generated names only when a device is communicating with another device and enable users to assign more user friendly names to be used in parallel.

Comparing previous options, current home networks and knowledge level of home users and thinking about what the IPv6-only home network might bring, it looks like the old way of using operating system given names or zero configuration networking are the two most viable manners to name devices at home.

9.1.2 IP address and name acquisition and services

Even if the naming might work in the same way in a legacy device as in a newly purchased device, the DNS and DHCP configuration will create challenges. It cannot be presumed that all legacy devices can be updated to support IPv6, so home users will have to update their old hardware. When this is done there should be no obstacles for home devices to connect and operate in IPv6 networks. DHCPv6 or other such a protocol can be used to provide home user devices an IPv6 address and a fully qualified host name that can be made reachable from the Internet if needed.

Having the target in automation, services provided by network and its hosts should be found using no more than a couple of clicks. Therefore a few service discovery protocols were examined. The first of them was DNS-SD, which is compatible with both conventional DNS and multicast DNS. Due to the compatibility it is preferred in some implementations, but downside is that the current protocol specification does not scale only outside the local network. Therefore it cannot be used in a wide area network without enhancements, which have been designed to overcome its limitations.

Other two service discovery protocols examined were SSDP and SLP. Neither has the same limitation that DNS-SD has, so they can be used in larger networks without modifications. Due to the fact that DNS is the name service used in the Internet, protocols incompatible with DNS cannot be used in service discovery if interaction is needed. The case is the same when choosing the name service protocol for a network.

When looking at the interaction of protocols, some kind of a drawback is that currently multicast DNS and conventional DNS do not have a standard manner to interact. It is not quite a show stopper, but it would ease a lot, if these two protocols could exchange information.

9.1.3 Connection with an ISP

From the ISP point of view the easiest way to enable Internet name service at home is to provide domain names independent from the ISP domain names to customers and bundle the home router into the Internet connection subscription. This way there will be no legal obligation to the ISP about the domain name in case of abuse.

When owning the home router the ISP can pre-configure secure name service connection between the home router it owns and its DNS servers. However, an issue may rise when tens or hundreds of thousands of customers are having name service at home. In this situation current security algorithms used between DNS servers may have to be changed since they require too much administrational work.

If home routers are owned by customers, ISP cannot be absolutely sure that a device communicating with its DNS servers is the one it claims to be. In that case there will be also a security issue, how to deliver required security parameters from ISP to the customer and be able to update them frequently. So a strong enough, but user friendly online authentication method would be needed to authenticate the customer.

When looking from the customer point of view, such a DNS topology should be used that hides the home network and its devices from the Internet, but enables local name service also during outages in the Internet connection. DNS traffic from the Internet should not be passed to the customer connection, but the ISP name server should answer to queries regarding the home network domain. This protects home network and its devices from certain attacks, although never protecting it thoroughly.

Summarizing this section, the easiest setup is when an ISP owns the home router. Home network name service can then be connected securely to the Internet DNS. It is the most care free implementation to a home customer as well.

9.1.4 Lab

The lab results were quite predictable. Already in the theory part it could be seen that although the subject is discussed in the IETF and such organizations, there is no standard for the whole package yet. This is also a good thing, because otherwise I would be repeating someone else's research. For the theory and lab I tried to search the Internet for a similar thesis or other research, but I did not find one. I know that there are working implementations of ISP provided IPv6 home networks in the world, but I am not aware of any of them implementing a name service.

The most significant difference that I found between protocols used in IPv4 and IPv6 networks during the lab in detail, was the lack of routing feature in the DHCPv6. This was not revealed in the theory part since I saw no reason to include DHCP in it. When reading through specifications I did lean to the party that is working to add routing feature to DHCPv6, although I do understand the reasons it was left out from the original specification. Other than that the lab results were quite predictable as said in the beginning. DHCPv6 needed another software running alongside in order to deliver routing information to clients.

Securing the connection between the home network and the ISP in the lab was easy, but in real life the situation is different. The authentication protocol used in the lab does not scale very well to the large ISP environments. Previously discussed possibility, where the ISP bundles the home router with the connection subscription, would make the implementation possible with the current authentication protocols.

9.2 Discussion

The results of the lab correspond very well to the information acquired in the theory part. Already when studying the theory, I got the impression that it is possible to implement such a network somehow. It would have been somewhat disappointing to discover that even though the theory for the implementation exists, there are not yet enough actual pieces to complete the puzzle.

An additional challenge during the writing process was given by the authors of the documents that I am referring to. Since many references were still in draft stage and were often modified, I had to check them every once in a while in case the newer version included valuable information.

At the moment there is no protocol to enable automatic host naming that could be connected to the Internet DNS. There are protocols for automatic naming and of course the protocol for the Internet name service, the DNS. Still there is a gap between these two, which should be filled somehow to enable seamless operation of name service in the home networks.

The connection to the Internet name service have to be made using DNS and since the interaction between DNS and automatic name service protocols is very poor, it looks like DNS have to be used in the home network too. However, it does not prevent using protocol such as DNS-SD alongside. Protocols can also complement each other like DHCPv6 and RA do.

My opinion is that the thesis accomplished well objectives assigned to it. If there had been more time, the lab part could have been fine-tuned further. Some protocol available at the moment set limits to the implementation and steer it to a certain direction. All in all the implementation is quite rough on the edges, but it is a good start if an ISP wants to process it further to a real live product that could be offered to customers.

10 CONCLUSIONS

During the research process it became clear that a lot has to be done before an ISP can offer name service to its home customers. Giving a name to a device is quite easy, but using the same name to connect to the device from the Internet is not so simple anymore. In a small network naming every device manually is not a problem, but in larger networks automatic naming process is vital. Protocols for automatic naming do exist, but they do not interact with the Internet DNS, so standardization is needed.

To make the home network as user friendly as possible the services should be easy to find. For this purpose there are protocols that can be used, or at least they are enabled and ready to use.

At the moment home users have to have advanced skills to enable name service at home and connect it to the Internet DNS, using the methods introduced in the thesis. There are already dynamic DNS services available, but they all are cloud services. In order to provide a service that remains operational during outages, name server has to be brought to the home network. Appropriate DNS topology is also vital to protect the home network from hazards of the Internet.

The current security situation in the Internet is not so good. Organizations having enough power and resources can use electronic surveillance to listen to practically anyone they like and need to. Luckily there are still some things that an individual can and should do. Securing name service connections can be used to provide some security through authentication against those who have some but not all resources to monitor the Internet traffic. Like organizations are securing their data transfer, a home user should do the same when transferring name service and other information over the Internet. Name service implementation should be built so that both customers and ISPs benefit from it. Allowing ISP to manage home router seems to be the best solution for both.

As always, the work is never completely done. There is always something that could be researched further. This thesis leaves also some areas open that need to be covered if an ISP would process the thesis further into a real product. For example, a customer service portal needs to be developed to serve customers when they have issues with DNS. A home router firmware could be tailored by an ISP to include only needed features for customers purchasing the DNS service. The finished product should offer an option that only a defined part of the home network hosts are visible to the Internet while others stay invisible.

If there would be enough interest for such a product, it should not take long when some partial setups could be implemented. Although the automatic naming seems to be the hardest part currently, there are multiple research projects working in this area.

REFERENCES

Aitchison, R. Pro DNS and BIND 10. [PDF]. Apress. 2011. Available at: http://it-ebooks.info/go.php?id=1350-1377289510-fbc97c99a9d6622479f4ff7b63f4a552.

Albitz, P., Cricket, L. DNS and BIND. 5th Edition. Sebastopol (CA) 2006, O'Reilly Media, Inc. 640 p.

Amberg, E. DNSSEC verheiratet Namensauflösung und PKI. [WWW]. 2008. [accessed on 14.9.2013].
Available at: http://www.linux-magazin.de/Ausgaben/2008/05/Beglaubigte-Adressen.

Arends, R., Austein, R., Larson, M., Massey, D., Rose, S. 2005. Resource Records for the DNS Security Extensions. IETF. RFC 4034. 29 p.
Available at: http://www.ietf.org/rfc/rfc4034.txt.

Atkins, D., Austein, R. 2004. Threat Analysis of the Domain Name System (DNS). IETF. RFC 3833. 16 p. Available at: http://www.ietf.org/rfc/rfc3833.txt.

Behringer, M., Pritikin, M., Bjarnason, S. 2012. Bootstrapping Trust on a Homenet. IETF draft [WWW]. 4 p. [accessed on 18.9.2013].
Available at: http://tools.ietf.org/id/draft-behringer-homenet-trust-bootstrap-00.txt.

Bhandari, S., Fajalia, B., Schmieder, R., Orr, S., Dutta, A. 2013. Extending multicast DNS across local links in Campus and Enterprise networks. IETF draft [WWW]. 12 p. [accessed on 17.9.2013]. Available at: http://tools.ietf.org/id/draft-bhandari-dnssdext-mdns-gateway-01.txt.

Cheshire, S., Aboba, B., Guttman, E. 2005. Dynamic Configuration of IPv4 Link-Local Address. IETF. RFC 3927. 33 p. Available at: http://www.ietf.org/rfc/rfc3927.txt.

Cheshire, S., Steinberg, D. Zero Configuration Networking: The Definitive Guide. Sebastopol (CA) 2005, O'Reilly Media. 254 p.

Cheshire, S. 2012. Zero Configuration Networking (Zeroconf) [WWW].
[accessed on 25.2.2013]. Available at: http://www.zeroconf.org/.

Cheshire, S. 2013. Hybrid Unicast/Multicast DNS-Based Service Discovery. IETF draft [WWW]. 9 p. [accessed on 16.8.2013]. Available at: http://tools.ietf.org/id/draft-cheshire-mdnsext-hybrid-02.txt.

Cheshire, S., Krochmal, M. 2013a. Multicast DNS. IETF. RFC 6762. 70 p. Available at: http://www.ietf.org/rfc/rfc6762.txt.

Cheshire, S., Krochmal, M. 2013b. DNS-Based Service Discovery. IETF. RFC 6763. 49 p. Available at: http://www.ietf.org/rfc/rfc6763.txt.

Chown, T., Arkko, J., Brandt, A., Troan, O., Weil, J. 2013. Home Networking Architecture for IPv6. IETF draft [WWW]. 47p. [accessed on 27.5.2013]. Available at: http://tools.ietf.org/id/draft-ietf-homenet-arch-08.txt.

Cloetens, W., Lemordant, P., Migault, D. 2012. IPv6 Home Network Naming Delegation Architecture. IETF draft [WWW]. [accessed on 25.7.2013]. Available at: http://tools.ietf.org/id/draft-mglt-homenet-naming-delegation-00.txt.

Cotton, M., Vegoda, L. 2010. Special Use IPv4 Addresses. IETF. RFC 5735. 11 p. Available at: http://www.ietf.org/rfc/rfc5735.txt.

Damas, J., Graff, M., Vixie, P. 2013. Extension Mechanisms for DNS (EDNS(0)). IETF. RFC 6891. 16 p. Available at: http://www.ietf.org/rfc/rfc6891.txt.

Davidowicz, D. Domain Name System (DNS) Security. [WWW]. 1999. [accessed on 23.8.2013]. Available at: http://compsec101.antibozo.net/papers/dnssec/dnssec.html.

Davison, W. 2013. Rsync [WWW]. [accessed on 12.7.2013]. Available at: http://rsync.samba.org/.

Dec, W., Mrugalski, T., Sun, T., Sarikaya, B., Matsumoto, A. 2012. DHCPv6 Route Options. . IETF draft [WWW]. [accessed on 24.10.2013]. Available at: http://tools.ietf.org/id/draft-ietf-mif-dhcpv6-route-option-05.txt.

Droms, R. Bound, J., Volz, B., Lemon, T., Perkins, C., Carney, M. 2003. Dynamic Host Configuration Protocol for IPv6 (DHCPv6) IETF. RFC 3315. 101 p. Available at: http://www.ietf.org/rfc/rfc3315.txt

Eastlake, D. 1999. Domain Name System Security Extensions. IETF. RFC 2535. 47 p. Available at: http://www.ietf.org/rfc/rfc2535.txt.

Eastlake, D. 2000a. DNS Request and Transaction Signatures (SIG(0)s) . IETF. RFC 2931. 10 p. Available at: http://www.ietf.org/rfc/rfc2931.txt.

Eastlake, D. 2000b. Secret Key Establishment for DNS (TKEY RR). IETF. RFC 2930. 16 p. Available at: http://www.ietf.org/rfc/rfc2930.txt.

EfficientIP. DNS architectures. [PDF]. EfficientIP. 2012. [Accessed on: 18.10.2013]. Available at: http://www.efficientip.com/docs/eipwebinar-dnsarchitectures-022013.pdf.

Ferguson, P., Senie, D. 2000. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. IETF. RFC 2827. 10p. Available at: http://www.ietf.org/rfc/rfc2827.txt.

Goland, Y. Y., Cai, T., Leach, P., Gu, Y., Albright, S. 1999. Simple Service Discovery Protocol/1.0 Operating without an Arbiter. . IETF draft [WWW]. [accessed on 16.8.2013]. Available at: http://tools.ietf.org/id/draft-cai-ssdp-v1-03.txt.

Gudmundsson, O., Koch, P. DNS (Domain Name System) Tutorial @ IETF-70 (DNS for protocol designers). [PDF]. IETF. 2.12.2007. [accessed on 22.8.2013]. Available at: http://www.ietf.org/proceedings/70/slides/dnstut-0.pdf.

Guttman, E., Perkins, C., Veizades, J., Day, M. 1999. Service Location Protocol, Version 2. IETF. RFC 2608. 54 p. Available at: http://www.ietf.org/rfc/rfc2608.txt.

Hinden, R., Deering, S. 2006. IP Version 6 Addressing Architecture. IETF. RFC 4291. 25 p. Available at: http://www.ietf.org/rfc/rfc4291.txt.

IANA. 2013a. IPv4 Multicast Address Space Registry. [WWW]. [accessed on 6.2.2013]. Available at: http://www.iana.org/assignments/multicast-addresses/multicast-addresses.txt.

IANA. 2013b. IPv6 Multicast Address Space Registry. [WWW]. [accessed on 6.2.2013]. Available at: http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.txt.

IANA. 2013c. Service Name and Transport Protocol Port Number Registry. [WWW] [accessed on 18.8.2013]. Available at: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt.

IANA. 2013d. Domain Name System (DNS) Parameters. [WWW]. [accessed on 22.8.2013].
Available at: http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml.

ICANN. 2013. DNSSEC Deployment Graph. [WWW]. [accessed on 28.8.2013]. Available at: http://www.icann.org/en/news/in-focus/dnssec/deployment-graph.

IETF. 2008. Zero Configuration Networking (zeroconf) [WWW]. [accessed on 27.2.2013]. Available at: http://www.ietf.org/wg/concluded/zeroconf.

ISC. 2013. The most widely used name server software: BIND. [WWW]. [accessed on 8.10.2013]. Available at: http://www.isc.org/downloads/bind/

ISC. BIND 9 Administrator Reference Manual. [PDF]. ISC. 2010. [accessed on 10.6.2013] Available at: http://www.bind9.net/arm97.pdf.

Kitamura, H., Ata, S. 2012. Corresponding Auto Names for IPv6 Addresses. IETF draft [WWW]. [accessed on 8.3.2013]. Available at: http://tools.ietf.org/id/draft-kitamura-ipv6-auto-name-03.txt.

Kelley, S. 2013. Dnsmasq. [WWW]. [accessed on 8.10.2013].
Available at: http://www.thekelleys.org.uk/dnsmasq/doc.html

Kolkman, O., Mekking, W., Gieben, R. 2012. DNSSEC Operational Practices, Version 2. IETF. RFC 6781. 71 p. Available at: http://www.ietf.org/rfc/rfc6781.txt.

Laurie, B., Sisson, G., Arends, R., Arends, B. 2008. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. IETF. RFC 5155. 52 p.
Available at: http://www.ietf.org/rfc/rfc5155.txt.

Lewis, L., Hoenes, A. 2010. DNS Zone Transfer Protocol (AXFR). IETF. RFC 5936. 29 p. Available at: http://www.ietf.org/rfc/rfc5936.txt.

Lynn, K., Sturek, D. 2012. Extended Multicast DNS. IETF draft [WWW] [accessed on 19.10.2013]. Available at: http://tools.ietf.org/html/draft-lynn-homenet-site-mdns-01.

Migault , D., Cloetens, W., Lemordant, P., Griffiths, C. 2012. IPv6 Home Network Front End Naming Delegation. IETF draft [WWW]. [accessed on 20.3.2013]. Available at: http://tools.ietf.org/id/draft-mglt-homenet-front-end-naming-delegation-01.txt.

Mockapetris, P. 1987. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. IETF. RFC 1035. 55p. Available at: http://www.ietf.org/rfc/rfc1035.txt.

Mrugalski, T. Dibbler – a portable DHCPv6 User's guide. [PDF]. 24.9.2013 [accessed on 18.10.2013]. Available at: http://klub.com.pl/dhcpv6/doc/dibbler-user.pdf.

Ohta, M. 1996. Incremental Zone Transfer in DNS. IETF. RFC 1995. 8 p. Available at: http://www.ietf.org/rfc/rfc1995.txt.

Presser, A., Farrell, L., Kemp, D., Lupton, W., Tsuruyama, S., Albright, S., Donoho, A., Ritchie, J., Roe, B., Walker, M., Nixon, T., Evans, C., Rawas, H., Freeman, T., Park, J., Chan. C., Reynolds, F., Costa-Requena, J., Ye, Y., McGee, T., Knapen, G., Bodlaender, M., Guidi, J., Heerink, L., Gildred, J., Messer, A., Kim, Y., Wischy, M., Fiddian-Green, A., Fairman, B., Tourzan, J., Fuller, J. UPnP™ Device Architecture 1.1. [PDF]. UPnP Forum. 15.10.2008. [accessed on 16.8.2013]
Available at: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf.

Thaler, D., Draves, R., Matsumoto, A., Chown, T. 2012. Default Address Selection for Internet Protocol Version 6 (IPv6) IETF. RFC 6724. 32 p.
Available at: http://www.ietf.org/rfc/rfc6724.txt.

Thomson, S., Narten, T., Jinmei, T. 2007. IPv6 Stateless Address Autoconfiguration. IETF. RFC 4862. 30 p. Available at: http://www.ietf.org/rfc/rfc4862.txt.

Ts, J., Eckstein, R., Collier-Brown, D. Using Samba [e-book]. 2nd Edition. Sebastopol (CA), O'Reilly Media, Inc. February 2003. [accessed on 19.10.2013]
Available at: http://www.samba.org/samba/docs/using_samba/toc.html.

Van de Velde, G., Hain, T., Droms, R., Carpenter, B., Klein, E. 2007. Local Network Protection for IPv6. IETF. RFC 4864. 36 p. Available at: http://www.ietf.org/rfc/rfc 4864.txt.

Veizades, J., Guttman, E., Perkins, C., Kaplan, S. 1997. Service Location Protocol. IETF. RFC 2165. 72 p. Available at: http://www.ietf.org/rfc/rfc2165.txt.

Wellington, B. 2000. Secure Domain Name System (DNS) Dynamic Update. IETF. RFC 3007. 9 p. Available at: http://www.ietf.org/rfc/rfc3007.txt.

Vixie, P. 1996. A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY). IETF. RFC 1996. 7 p. Available at: http://www.ietf.org/rfc/rfc1996.txt.

Vixie, P., Gudmundsson, O., Eastlake 3rd, D., Wellington, B. 2000. Secret Key Transaction Authentication for DNS (TSIG). IETF. RFC 2845. 15 p.
Available at: http://www.ietf.org/rfc/rfc2845.txt.

Vixie, P., Thomson, S., Rekhter, Y., Bound, J. 1997. Dynamic Updates in the DomainName System (DNS UPDATE). IETF. RFC 2136. 26 p.
Available at: http://www.ietf.org/rfc/rfc2136.txt.

Weiler, S. 2007. DNSSEC Lookaside Validation (DLV). IETF. RFC 5074. 11 p. Available at: http://www.ietf.org/rfc/rfc5074.txt.

Woodcock, B., Manning, B. 2000. Multicast Domain Name Service. IETF draft [WWW]. [accessed on 6.2.2013]. Available at: http://tools.ietf.org/id/draft-manning-dnsext-mdns-00.txt.

Woodyatt, J. 2011. Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service. IETF. RFC 6092. 36 p. Available at: http://www.ietf.org/rfc/rfc6092.txt.

APPENDIX 1: SOFTWARE VERSIONS

ISP DNS server – a workstation PC
OS: Ubuntu 12.04.3
DNS software: BIND 9.8.1-P1

Homenet router 1 – Asus WL-500g Premium
OS: Openwrt 12.09
DNS software: BIND, version 9.9.1-P3-1
DHCP software: Dibbler DHCPv6 server, version 0.8.2

Homenet router 2 – a workstation PC
OS: Ubuntu 12.04.3
DNS software: BIND 9.8.1-P1
DHCP software: Dibbler DHCPv6 server, version 1.0.0RC1
RA software: RADVD, version 1.8.3

Home user device 1 – a laptop PC
OS: Ubuntu 12.04.3
DHCP software: Dibbler DHCPv6 client, version 0.7.3

Home user device 2 – a laptop PC
OS: Lubuntu 12.04
DHCP software: Dibbler DHCPv6 client, version 0.8.2

Home user device 3 – a laptop PC
OS: Windows XP
DHCP software: Dibbler DHCPv6 client, version 0.8.4

APPENDIX 2: BIND CONFIGURATIONS

Configuration files shared by the ISP and the homenet DNS server

/etc/bind/named.conf

```
// This is the primary configuration file for the BIND DNS server named.
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
logging{
        channel normal_log {
                file "/var/log/named/normal.log";
                severity error;
                print-category yes;
                print-severity yes;
                print-time yes;
                };
        channel security_log {
                file "/var/log/named/security.log";
                severity info;
                print-category yes;
                print-severity yes;
                print-time yes;
                };
        channel dnssec_log {
                file "/var/log/named/dnssec.log";
                severity debug 3;
                print-category yes;
                print-severity yes;
                print-time yes;
                };
        category dnssec { dnssec_log;};
        category security { security_log;};
        category default {normal_log;};
};
```

/etc/bind/named.conf.options

```
options {
        directory "/var/cache/bind";
        version "not currently available";
        dnssec-validation auto;
        auth-nxdomain no;    # conform to RFC1035
         listen-on-v6 { any; };
        allow-transfer {none;};
        allow-update {none;};
};
```

Homenet router DNS configuration files

/etc/bind/named.conf.local

```
// TSIG key for us.example.com
key "us.example.com" {
      algorithm hmac-md5;
      secret "yWiWE3ryhgIUHWs7pbifHQ==";
      };
// Slave DNS server for us.example.com
server 2001:db8:1::1 { keys {"us.example.com";};};
// Zone configurations
zone "us.example.com" {
      type master;
      file "/var/lib/bind/us.example.com.hosts";
      allow-update { 2001:db8:0:2::1;};
      allow-transfer { key "us.example.com";};
      also-notify { 2001:db8:1::1;};
      notify yes;
      };
zone "2.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa" {
      type master;
      file "/var/lib/bind/2001:db8:0:2::_64.rev";
      notify yes;
      allow-update { 2001:db8:0:2::1;};
      allow-transfer { key "us.example.com";};
      also-notify { 2001:db8:1::1;};
      };
```

/var/lib/bind/us.example.com.hosts

```
$ORIGIN us.example.com.
$TTL 38400      ; 10 hours 40 minutes
@       IN SOA  ns.us.example.com. hostmaster.us.example.com. (
        1500000000 ; serial
        10800      ; refresh (3 hours)
        3600       ; retry (1 hour)
        604800     ; expire (1 week)
        38400      ; minimum (10 hours 40 minutes)
        )
        NS      ns.us.example.com.
ns      AAAA    2001:db8:1::1
```

/var/lib/bind/2001:db8:0:2::_64.rev

```
$ORIGIN 2.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.
$TTL 38400      ; 10 hours 40 minutes
@       IN      SOA ns.us.example.com. hostmaster.us.example.com. (
        1500000000 ; serial
        10800      ; refresh (3 hours)
        3600       ; retry (1 hour)
        604800     ; expire (1 week)
        38400      ; minimum (10 hours 40 minutes)
        )
        NS      ns.us.example.com.
```

ISP DNS server configuration files

/etc/bind/named.conf.local

```
// TSIG keys for example.com and us.example.com
key "example.com" {
      algorithm hmac-md5;
      secret "furBF003k1mMpgQUEodUSw==";
      };
key "us.example.com" {
      algorithm hmac-md5;
      secret "yWiWE3ryhgIUHWs7pbifHQ==";
      };
// Slave DNS server for example.com
server 2001:db8:2::1 {
      keys {"example.com";};
};
// Master DNS server for us.example.com
```

```
server 2001:db8:1::2 {
        keys {"us.example.com";};
};
// Zone configurations
zone "example.com" {
        type master;
        file "/var/lib/bind/example.com.hosts";
        allow-transfer {key "example.com";};
            notify yes;
        };
zone "0.0.0.0.1.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa" {
        type master;
        file "/var/lib/bind/2001:db8:1:0::_64.rev";
        allow-transfer {key "example.com";};
            notify yes;
        };

zone "us.example.com" {
        type slave;
        masters {2001:db8:1::2;};
        file "/var/lib/bind/us.example.com.hosts";
        allow-notify {key "us.example.com";};
        };
zone "2.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa" {
        type slave;
        masters {2001:db8:1::2;};
        file "/var/lib/bind/2001:db8:0:2::_64.rev";
        allow-notify {key "us.example.com";};
        };
```

/var/lib/bind/example.com.hosts

```
$TTL 38400
$ORIGIN example.com.
 @      IN      SOA     ns.example.com. hostmaster.example.com. (
            1500000000 ; serial
            10800      ; refresh (3 hours)
            3600       ; retry (1 hour)
            604800     ; expire (1 week)
            38400      ; minimum (10 hours 40 minutes)
        )
        IN      NS      ns.example.com.
        IN      NS      ns.example.net.
```

```
us    IN    NS      ns.example.com.
ns    IN    AAAA    2001:DB8:1::1
```

/var/lib/bind/2001:db8:0:2::_64.rev

```
$TTL 38400
$ORIGIN 1.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.
@       IN      SOA     ns.example.com.       hostmaster.example.com. (
        1500000000 ; serial
        10800      ; refresh (3 hours)
        3600       ; retry (1 hour)
        604800     ; expire (1 week)
        38400      ; minimum (10 hours 40 minutes)
        )
@       IN      NS      ns.example.com.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.    IN    PTR    ns.example.com.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.    IN    PTR    ns.us.example.com.
```

APPENDIX 3: HOMENET ROUTER CONFIGURATION FILES

Dibbler DHCPv6 server configuration

/etc/dibbler/server.conf

```
# Logging level, 8 = debug
log-level 8

# set preference of this server to 0 (higher = more prefered)
preference 0

# Set protocol to one of the following values: udp, tcp, any
ddns-protocol udp

# Sets DDNS Update timeout (in ms)
ddns-timeout 1000

# specify address of DNS server to be used for DDNS
fqdn-ddns-address 2001:db8:0:2::1

iface "eth1" {

  t1 1800-2000
  t2 2700-3000
  prefered-lifetime 3600
  valid-lifetime 7200

# assign addresses from this class
 class {
   pool 2001:db8:0:2::/64
 }

# default route for clients
 next-hop 2001:db8:0:2::1

# provide DNS server location to the clients
# also server will use this address to perform DNS Update,
# so it must be valid and DNS server must accept DNS Updates.
```

```
 option dns-server 2001:db8:0:2::1

# provide their domain name
 option domain us.example.com

# provide fully qualified domain names for clients
# First parameter defines, who will do the updates (0 - nobody, 1 - server will
# do PTR, client will do AAAA, 2 - AAAA and PTR done by server)
# Third parameter specified reverse zone length
option fqdn 2 64
        spare1.us.example.com,
        spare2.us.example.com,
        spare3.us.example.com

# specify what to do with client's names that are not on the list
# 0 - reject
# 1 - send other name from allowed list
# 2 - accept any name client sends
# 3 - accept any name client sends, but append specified domain suffix
# 4 - ignore client's hint, generate name based on his address, append domain name

 accept-unknown-fqdn 3 us.example.com
 }
```

Router Advertisement configuration

/etc/radvd.conf

```
interface eth1
{
        AdvManagedFlag on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        AdvSendAdvert on;
  prefix 2001:db8:1234:4321::/112
  {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
  };
};
```

APPENDIX 4: HOMENET CLIENT CONFIGURATION FILES

Dibbler DHCPv6 client configuration file

/etc/dibbler/client.conf

```
# Script e.g. to update IPv6 routing table
script "/var/lib/dibbler/client-notify.sh"

# Logging level, 8 = debug
log-level 8

iface eth0 {
# ask for address
   ia

# ask for DNS server's address, domain and routing information
   option dns-server
   option domain
   routing 1

# provide a fully qualified domain name hint for the server
  option fqdn fujitsutooot.us.example.com
}
```

Dibbler DHCPv6 client notify script

/var/lib/dibbler/client-notify.sh

```
#!/bin/bash
 LOGFILE=/var/lib/dibbler/client.sh-log
 # uncomment this to get full list of available variables
set >> $LOGFILE
 echo "-----------" >> $LOGFILE
 if [ "$OPTION_NEXT_HOP" != "" ]; then
    ip -6 route del default > /dev/null 2>&1
   ip -6 route add default via ${OPTION_NEXT_HOP} dev $IFACE
   echo "Added default route via ${OPTION_NEXT_HOP} on interface $IFACE/$IFINDEX" >> $LOGFILE
 fi
 if [ "$OPTION_NEXT_HOP_RTPREFIX" != "" ]; then
    NEXT_HOP=`echo ${OPTION_NEXT_HOP_RTPREFIX} | awk '{print $1}'`
   NETWORK=`echo ${OPTION_NEXT_HOP_RTPREFIX} | awk '{print $2}'`
   #LIFETIME=`echo ${OPTION_NEXT_HOP_RTPREFIX} | awk '{print $3}'`
   METRIC=`echo ${OPTION_NEXT_HOP_RTPREFIX} | awk '{print $4}'`
    if [ "$NETWORK" == "::/0" ]; then
       ip -6 route del default > /dev/null 2>&1
      ip -6 route add default via ${OPTION_NEXT_HOP} dev $IFACE
      echo "Added default route via  ${OPTION_NEXT_HOP} on interface
$IFACE/$IFINDEX" >> $LOGFILE
    else
       ip -6 route add ${NETWORK} nexthop via ${NEXT_HOP} dev $IFACE weight
${METRIC}
      echo "Added nexthop to network ${NETWORK} via ${NEXT_HOP} on interface
$IFACE/$IFINDEX, metric ${METRIC}" >> $LOGFILE
    fi
fi
 if [ "$OPTION_RTPREFIX" != "" ]; then
    ONLINK=`echo ${OPTION_RTPREFIX} | awk '{print $1}'`
   METRIC=`echo ${OPTION_RTPREFIX} | awk '{print $3}'`
   ip -6 route add ${ONLINK} dev $IFACE onlink metric ${METRIC}
   echo "Added route to network ${ONLINK} on interface $IFACE/$IFINDEX onlink,
metric ${METRIC}" >> $LOGFILE
fi
 if [ "$ADDR1" != "" ]; then
   echo "Address ${ADDR1} (operation $1) to client $REMOTE_ADDR on inteface
$IFACE/$IFINDEX" >> $LOGFILE
fi

if [ "$PREFIX1" != "" ]; then
   echo "Prefix ${PREFIX1} (operation $1) to client $REMOTE_ADDR on inteface
$IFACE/$IFINDEX" >> $LOGFILE
fi
 # sample return code. Dibbler will just print it out.
exit 3
```