Securing the transition to IPv6 in small/medium sized businesses

Henrik Dalbakk

Masteroppgave
Master i informasjonssikkerhet
30 ECTS
Avdeling for informatikk og medieteknikk
Høgskolen i Gjøvik, 2010

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Securing the transition to IPv6 in small/medium sized businesses

Henrik Dalbakk

1st July 2010

Abstract

The Internet has evolved in a frantic pace the last 15 years, few could anticipate the rapid growth and the enormous impact it has had on global communication. When something evolved this fast it is bound to get into problems, so has the Internet. The protocol used for communication today, IPv4, is running out of addresses, and it is doing it fast. When this becomes a reality we face the issue of not being able to grant new participants access to the network. Fortunately the replacement for IPv4 has been around for quite some time, namely IPv6. IPv6 does not have the same shortcomings as IPv4 and will allow an almost infinite set of participants.

Existing networks that run IPv4 today will because of this require an upgrade to be able to use the additional protocol and communicate with both IPv4 and the IPv6 hosts. This means that network administrators will have to open up a new road into their network, and effectively double the entry points. In this thesis we have looked at how owners of small and medium sized businesses can do this in a secure fashion. We have enumerated and analysed the most suitable transition technologies and chosen a subset that are added to a test network. The experiences we drew from these experiments are presented in such a fashion that it will give implementers a set of recommendations for best practice when it comes to making the implementation of IPv6 a success.

Sammendrag

Internett har utviklet seg i et rasende tempo de siste 15 årene. Få kunne forvente den raske veksten og den enorme effekten det har hatt på global kommunikasjon. Når noe vokser i et slikt tempo er det nødt til å oppstå voksesmerter, dette er også tilfelle med internett. Protokollen som brukes for kommunikasjon i dag, IPv4, er i ferd med å gå tom for adresser, og det gjør den fort. Når dette blir en realitet vil vi stå ovenfor problemet at vi ikke er i stand til å gi nye aktører tilgang til nettverket. Heldigvis er erstatningen for IPv4 vært klar en stund, nemlig IPv6. IPv6 har ikke de samme manglene som IPv4 og vil gi mulighet for et nesten uendelig sett av deltakere.

Eksisterende nettverk som kjører IPv4 i dag, vil på bakgrunn av dette kreve en oppgradering for å kunne bruke denne nye protokollen og kommunisere med både IPv4 og IPv6-klienter. Dette betyr at nettverksadministratorer vil måtte åpne en ny vei inn i sitt nettverk, og praktisk talt fordoble antallet ”innganger”. I denne oppgaven har vi sett på hvordan eierne av små og mellomstore bedrifter kan gjøre dette på en sikker måte. Vi har gått gjennom og analysert de mest egnede overgangsteknologiene og valgt ut et sett med teknikker som er implementert i et test-nettverk. Erfaringene vi trakk fra disse eksperimentene er analysert, diskutert og presenteres på en slik måte at det vil gi administratorer et godt grunnlag for å bestemme seg for en implementeringsstrategi for IPv6 i sitt nettverk.

Acknowledgements

When writing a master thesis you feel kind of overwhelmed with work at times and on many occasions wonder how on earth it is going to be finished in time. There are people behind the name on the front page that have helped in getting to the finish line and they deserve a special thanks. First and foremost I would like to thank my supervisor Lasse Øverlier who was enthusiastic about the topic at hand, helped me throughout the process and even let me borrow personal server grade hardware for my experiments.

A special thanks also goes to my opponent Anders Orsten Flaglien, his comments and remarks has been a tremendous help in seeing the flaws of my thesis and pointing out all the little things that you get blind for when reading your own work. Other people earning special mention is Jon Langseth for his continuous willingness to discuss problems concerning the work, and also all other topics we have covered the last six months. Trond Endrestøl at Gjøvik Technical College for giving me an external native IPv6 shell for testing.

This thesis has also demanded access to the infrastructure here at Gjøvik University College and the always helpful people at the IT-department deserves a big thanks. Not only for their willingness to help facilitate the space needed for the hardware and configuring the network to suit my needs. But also a big thanks for all the interesting discussions, technical advisory, free coffee, the occasional cake and the genuine inviting atmosphere making procrastination easy as pie.

Julie Hagen Helland deserves a medal for her patience and acceptance of the time spent doing this thesis, and when I did have time off, for helping me take my mind of it and relax. My mother May Tove Dalbakk and my father Jonny Dalbakk for always keeping my spirit up and telling me that it was going to go okay, even in times when motivation was low and the anxiety for various deadlines were in full effect.

Last but not least are all the people at the masters lab at GUC, we have shared six months filled with thesis anxiety, heavy coffee drinking, LaTeX head scratching, rooftop ice creams and had a generally good time. This last part of my time at GUC will not be forgotten and looked back at with fond memories. To everybody mentioned here, and everybody I forgot; thank you!

Henrik Dalbakk 1st July 2010

Contents

Abstract . . . . . **iii**
Sammendrag . . . . . **v**
Acknowledgements . . . . . **vii**
Contents . . . . . **ix**
List of Figures . . . . . **xiii**
List of Tables . . . . . **xv**
List of Abbreviations . . . . . **xvii**
1 Introduction **1**
1.1 Topics covered by the project 1
1.2 Keywords 1
1.3 Problem description 2
1.4 Justification, motivation and benefits 2
1.5 Research questions 3
1.6 Research methods 4
1.7 Contributions 4
1.8 Chapter layout 5
2 Introduction to IPv6 . . . . . **7**
2.1 Address notation 7
2.2 Address types 7
2.2.1 Unicast 8
2.2.2 Multicast 9
2.2.3 Anycast 9
2.3 Autoconfiguration 9
3 Related work **11**
3.1 Why we should switch 11
3.1.1 Addresse space 11
3.1.2 End to end connectivity 12
3.1.3 Routing 13
3.1.4 Security 14
3.1.5 Controlling the unknown 15
3.2 Transition tools 15
3.2.1 Dual stack 16
3.2.2 Tunneling 16
3.2.3 Translation 17
3.2.4 Comparison 18
3.3 Experience and performance 18

3.4 Security considerations . . . 19
4 Transition mechanisms . . . 21
4.1 Dual stack . . . 21
4.1.1 Security considerations . . . 22
4.2 Tunneling . . . 23
4.2.1 Obsoleted tunneling mechanisms . . . 24
4.2.2 Configured tunnels . . . 25
4.2.3 6to4 . . . 27
4.2.4 Intra-site automatic tunneling addressing protocol (ISATAP) . . . 29
4.2.5 Teredo . . . 31
4.3 Protocol translation . . . 34
4.3.1 Network translators . . . 34
4.3.2 Host based translators . . . 36
4.4 Summary . . . 39
4.5 Selected techniques . . . 40
5 Planning the implementation . . . 41
5.1 Building a baseline network . . . 41
5.1.1 Physical hardware . . . 41
5.1.2 Network . . . 41
5.1.3 Software . . . 43
5.2 Implementation planning . . . 44
5.2.1 Implementing 6to4 . . . 46
5.2.2 Implementing ISATAP . . . 48
5.2.3 Implementing Teredo . . . 50
6 Experiments . . . 53
6.1 6to4 transition . . . 53
6.1.1 Firewall (pre-servers) . . . 53
6.1.2 Servers . . . 54
6.1.3 Firewall (post-servers) . . . 54
6.1.4 Clients . . . 55
6.1.5 Testing and Summary . . . 55
6.2 ISATAP transition . . . 56
6.2.1 Firewall . . . 57
6.2.2 ISATAP router . . . 57
6.2.3 ISATAP clients . . . 58
6.2.4 Testing and Summary . . . 58
6.3 Teredo transition . . . 59
6.3.1 Teredo clients . . . 59
6.3.2 Testing and Summary . . . 60
7 Discussion . . . 63
7.1 Doing the analysis . . . 63
7.2 6to4 . . . 63

7.2.1 LAN alterations . . . 63
7.2.2 WAN alterations . . . 64
7.2.3 Firewall alterations . . . 66
7.2.4 General 6to4 considerations . . . 67
7.3 ISATAP . . . 68
7.3.1 LAN alterations . . . 68
7.3.2 WAN alterations . . . 69
7.3.3 Firewall alterations . . . 70
7.3.4 General ISATAP considerations . . . 71
7.4 Teredo . . . 72
7.4.1 LAN alterations . . . 72
7.4.2 WAN alterations . . . 73
7.4.3 Firewall alterations . . . 73
7.4.4 General Teredo considerations . . . 74
7.5 Summary . . . 75
8 Conclusion . . . 77
9 Future work . . . 79
Bibliography . . . 81
A Configuration files . . . 89
A.1 m0n0wall . . . 89
A.1.1 m0n0wall IPv4 only . . . 89
A.1.2 m0n0wall IPv4 and IPv6 . . . 92
A.1.3 m0n0wall 6to4 . . . 96
A.1.4 m0n0wall ISATAP . . . 101
A.2 Linux configuration files . . . 107
A.2.1 Debian static configuration 6to4 . . . 107
A.2.2 Ubuntu 10.04 ISATAP router . . . 108

List of Figures

1 Address types 8
2 Generating interface ID 9
3 Internet users per 100 inhabitants 1997-2007 12
4 IPv6 packets encapsulated in IPv4 24
5 Configured tunnel 26
6 6to4 network flow 28
7 ISATAP configuration 30
8 Teredo tunnel setup 33
9 Original IPv4 UDP traffic 34
10 Original IPv4 UDP traffic analysed as IPv6 34
11 BIS diagram 37
12 BIA diagram 38
13 Baseline network hardware setup 43
14 Baseline network IP setup 45
15 The network with 6to4 implemented 56
16 The network with ISATAP implemented 59
17 The resulting network with Teredo 61

List of Tables

1 Address examples 7
2 Unicast variants 8
3 Autoconfiguration flags 10
4 Uninett announcements 13
5 Example on a Teredo IPv6 address 32
6 Summary of transition mechanisms 39
7 Physical hardware 41
8 m0n0wall setup 42
9 Virtual machines 44
10 6to4, tests performed 56
11 ISATAP, tests performed 58
12 Teredo, tests performed 61
13 6to4 routing rules 66
14 ISATAP router rules 70
15 Summary of tested transition mechanisms 75

List of Abbreviations

IPv6	=	Internet Protocol version 6
IPv4	=	Internet Protocol version 4
RFC	=	Request for Comments
IETF	=	Internet Engineering Task Force
ARPANET	=	Advanced Research Projects Agency Network
TCP/IP	=	Transmission Control Protocol/Internet Protocol
NAT	=	Network Address Translation
CIDR	=	Classless Inter Domain Routing
ARIN	=	American Registry for Internet Numbers
IEEE	=	Institute of Electrical and Electronics Engineers
IANA	=	The Internet Assigned Numbers Authority
DHCP	=	Dynamic Host Configuration Protocol
MAC	=	Media Access Control
NIC	=	Network Interface Card
RS	=	Router Solicitation
RA	=	Router Advertisement
DAD	=	Duplicate Address Detection
ICANN	=	Internet Corporation for Assigned Names and Numbers
ITU	=	International Telecommunication Union
ISP	=	Internet Service Provider
RIR	=	Regional Internet Registrars
LIR	=	Local Internet Registry
IPsec	=	Internet Protocol Security
ISATAP	=	Intra-Site Automatic Tunneling Addressing Protocol
UDP	=	User Datagram Protocol
NAT/PT	=	Network Address Translation/Protocol Translation
DSTM	=	Dual Stack IPv6 Dominant Transition Mechanism
DNS	=	Domain Name System
RTT	=	Round-Trip Time
ICMP	=	Internet Control Message Protocol
PKI	=	Public-Key Infrastructure
CPU	=	Central Processing Unit
LAN	=	Local Area Network
POP	=	Point of Presence
DoS	=	Denial of Service
PRL	=	Potential Router List
SIIT	=	Stateless IP/ICMP translation algorithm
TTL	=	Time To Live
ToS	=	Type of Service
DNS-ALG	=	DNS Application Layer Gateway
TRT	=	Transport Relay Transporter
DNSSEC	=	Domain Name System Security Extensions
SSH	=	Secure Shell

BIS	=	Bump In the Stack
BIA	=	Bump In the API
API	=	Application Programming Interface
HP	=	Hewlett-Packard
VM	=	Virtual Machine
WAN	=	Wide Area Network
AYIYA	=	Anything In Anything
LTS	=	Long Term Support
VLAN	=	Virtual LAN
IDS	=	Intrusion Detection System
AS	=	Autonomous System

1 Introduction

In this chapter we will give an introduction to the topics that are covered by this thesis and give a description of the problem as seen by the author. We will look into the motivation behind this work, list the research questions, give some insight into the methods used and list the contributions this work will give back to the field of study.

1.1 Topics covered by the project

This thesis is about how the adding of IPv6 to existing IPv4 networks can be solved in small or medium sized networks. We will evaluate different choices that can be made in realising this, pick a subset, make a baseline network with typical characteristics found in such networks, plan the implementations and execute them. After doing this we have discovered implementation problems and security challenges that stem from the different choices that can be made. These issues will be discussed in such a manner that we get the complete understanding of the alterations the network has undergone.

IPv6 is not a new protocol, the original document is RFC2460 [1] from 1998 and were made this early as the IETF already then saw the problems coming in IPv4. The protocol is meant as a successor to IPv4 and has differences in functionality that makes it non-compatible with the existing IPv4 network. To understand techniques used to make the two cooperate we first have to understand IPv6. This thesis contains a brief introduction to IPv6, not nearly everything, but enough for the reader to be able to follow the thesis. For readers wanting a more thorough explanation of IPv6, we can recommend the books ”IPv6 Essentials” [2] and ”IPv6 Network Administration” [3].

At a longer time perspective we should eventually completely move over to IPv6, this switch is however not going to be something that happens overnight and the two protocols will most likely coexist for quite some time. This coexistence is generating some new problems that network administrators need to take seriously and deal with in an appropriate manner to be able to implement IPv6 in a desirable fashion.

Little or no research has been done on the evaluation of IPv6 implementation scenarios for small or medium sized networks. The work found has been very focused on the security in IPv6 as a protocol [4, 5, 6], enumeration of the possible transitional technologies available [7, 8], theoretical analysis of specific transition mechanisms [9, 10] or scenarios/experience in large networks [11, 12, 13].

1.2 Keywords

Small/Medium sized network, Networking, IPv6 Implementation, IPv4 Networks, 6to4, ISATAP, Teredo, Firewall

1.3 Problem description

The predecessors to Internet as we know it today was founded on the idea that we would benefit from a network of interconnected hosts that would allow information to flow easily between every participant. This idea started the development of the ARPANET and the TCP/IP protocol, making way for what we today know as the Internet. When the Internet Protocol (IP) was invented, the address space was considered vast and enough for the foreseeable future. As history has done many times, we were proven wrong. The number of hosts on the Internet exploded and the number of available IPv4 addresses started to shrink. To return to the original ideas of an Internet that enables end-to-end connections many people think that the solution is IPv6, with the address space of 2^{128} possible addresses we should be able to retire the techniques that brakes connectivity.

While IPv6 is generally looked at as the desired solution for the problem, the implementation is going forward very slowly. A big part of the slow adoption of IPv6 is due to the fact that ISPs are sitting on the fence waiting for other providers to take the first step. This has gotten us in a ”chicken or the egg” situation. There are very few providers that offer native IPv6 connectivity to their customers because of the fact that demand is not high enough. The demand on the other hand can be said to stem from lack of availability. This is forcing network administrators that want to deploy it to look at alternative methods of acquiring IPv6 connectivity.

These alternative methods is what people in the networking industry is calling transition technologies, this is mechanisms that are inserted into the running network and can on different levels help facilitate the cooperation between the two IP versions. The problem is however the share abundance of techniques available and their different way of attacking the problem, which to choose?

The somewhat locked situation is putting the administrators of small, medium sized networks in an awkward position. They will benefit from implementing IPv6 in a secure fashion with proper knowledge and planning before execution, but because of their size they do not have the leverage to demand native connectivity from their ISP. Thus may forced to wait until the very end before it is suddenly available and should be implemented ”yesterday”, giving them the disadvantage of implementing new unfamiliar technology very fast and heightening the likelihood of errors and security flaws. This in spite the fact that businesses with such networks are a big portion of the businesses registered and should be important customers for the ISPs. As of January 2010 [14], 96% of all businesses with registered employees in Norway had under 50 people on the staff and are well in scope of this thesis.

1.4 Justification, motivation and benefits

In the international community today people are starting to address a problem they call the ”digital divide” [15], the gap between the developed and the developing world when it comes to, among other things, Internet connectivity. While the free addresses available today are being depleted rapidly, and the developing countries are equally rapidly establishing new connections, we will have a big problem ahead of us with the fact that IPv4 will be out of free addresses. Richard Jimmerson addressed this fact at the ARIN (American Registry for Internet Numbers) policy meeting held in April 2010[16]. His statement was:

> ... sometime between the middle and end of next year, receive a request for IPv4 address space

> that is justified and meets the policy. However, ARIN won't have the address space. So we'll have to say no for the very first time.

When we reach this point, new connections to the IPv4 Internet are just not possible. Although there are no new addresses to give out it is likely that IPv4 and IPv6 will live side by side for many years on forward. The planning for implementing IPv6 into our networks should by this fact start sooner rather than later. This will ensure that we do not segregate the world into IPv4 and IPv6 segments but get closer to the original idea that we should be world connectable and not inhibit new users from connecting. The avoidance of this scenario should be justification enough for ISPs to start the work needed to implement IPv6, a segregated Internet is not in anyone's interest and the consequence for service providers will be lost business.

From a security perspective this also means that we have to allow, for probably quite some time, traffic on both protocols simultaneously, and by this effectively doubling the entry points for potential attackers. Before doing this we need to consider the different mechanisms we have and evaluate how they will affect our existing network. This to make sure the extra functionality IPv6 will give us can be delivered in a secure fashion and with minimal extra risk for compromising functionality, availability and security. These processes is time consuming and doing them with a limited time frame can result in important aspects being overlooked. The added assurance we get from not being stressed on time should give the decision takers the incentive they need to devote the necessary resources.

Enabling IPv6 will make existing IPv4 networks connectable from both worlds, and not missing out of business opportunities from users residing in IPv4 or IPv6 only networks. With staying only in one domain like IPv4, you will most likely get traffic from IPv6 networks but maybe lack features IPv6 host are able to utilise. The Committee on Communications Policy of IEEE-USA[17] compared the switch between IPv4 and IPv6 to the switch between analogue and digital telephony lines, the analogue users could for a long time receive and place calls but would lack the extra functionality added by the new digital lines. This analogy is pretty precise when we think of how new technology always inspires new usage and services, and will most likely also be the case with IPv6 when it is getting widespread.

1.5 Research questions

The main question we will answer is: How do we add IPv6 to a small, medium sized IPv4 network while not sacrificing security, availability and functionality in the process? When trying to give an answer to this we will use the following questions:

1. Which different transitional techniques exist and who are suitable for ”our” network?
2. How do we plan and execute the transition in a secure fashion?
3. How will this affect the different parts of the running IPv4 network? To answer this we have looked at:
   - How the internal/external functionality is affected?
   - How the network security is affected?
   - How the offered services are affected?

4. To what extent can we conclude in a ”best practice” based on the practical implementation?

1.6 Research methods

In order to find answers and draw some conclusions from the posed research questions we needed to do an extensive literature study on which we could base our work. When doing the initial literature study we realised that getting a good understanding of IPv6 was paramount for us to be able to do this assignment, but also for the reader to follow it. This resulted in the crash course in IPv6 found in Chapter 2 which lays down the basics understanding to make the reader able to follow the thesis.

With the basics understanding covered we did an analysis of the different transitional techniques that were available to us when implementing IPv6. To narrow it down to a manageable set we looked at the different characteristics of the techniques and chose a set of three techniques that all had different scopes of operation, but would help in adding IPv6 to the network and were supported on the most common operating systems. This to analyse the ones most likely to be used and met ”in the wild”.

The techniques chosen in our survey of mechanisms had to get the same base network in which they were to be implemented. To make sure the mechanisms is assessed on a similar basis and the results can be reproduced and compared the network was set up in such a fashion that it would replicate the fundamental characteristics of the network a small, medium sized company would run.

With the baseline network in place we decided to do the implementation in two separate steps to simulate the process it should done in the real world. The first step is the detailed planning of the execution. When doing this planning we have employed a popular technique for complex problems, the divide and conquer algorithm as it is described by Knuth [18]. He describes the basic algorithm as a way of splitting complex problems into smaller ones until you get a problem that are manageable. This enables us to make plans for smaller parts of the network and make sure that they should work before the execution itself is performed and practical problems/experiences are collected for each transitional technique.

When the practical part of the thesis is performed we had an abundance of results and experiences from our experiments and needed to assess the impact it had on the network. In order to do this we chose to do a qualitative analysis [19] were we observed the changes in the networks and discussed the changes being made in an effort to make the difference in operation from before implementation was clear. We also summarised the findings and drew conclusions based on the findings, making us able us to give a detailed resumè of the pros and cons of the set.

1.7 Contributions

The thesis aim to contribute to the field of study in the following ways:

- A qualitative study on the implications of adding IPv6 to a typical small medium sized business network
  - Showing the importance of proper planning and giving an example of using ”divide and conquer” to create smaller segments.

  - A way to execute the plan in a controlled manner and add testing to detect errors as early as possible.

- An example implementation of three different transition techniques
- A recommendation for transition technique(s) based on practical implementation and analysis
- Added knowledge to the importance of rising the general IPv6 awareness

1.8 Chapter layout

This thesis has been divided into the following chapters to reflect the overall workflow of the project:

Chapter 1 - Introduction

In this first chapter we have focused on giving the reader a broad view of the problem at hand, why this is a problem and how we will investigate it.

Chapter 2 - Introduction to IPv6

IPv6 is new to many people and we have devoted this chapter to give the reader a crash course in a minimum of IPv6 so they are able to follow the thesis through.

Chapter 3 - Related work

This chapter consists of background material that have been done before in the research field.

Chapter 4 - Transition mechanisms

There exists a lot of different transition mechanisms that aim to in different ways to help us implement IPv6. This chapter will look at the most common ones and choose a set we find most suitable.

Chapter 5 - Planning the implementation

When we in the previous chapter has chosen the mechanisms we want to investigate, we now will explain the baseline network that they have been tested on and perform a detailed planning of how we want to deploy the different mechanisms.

Chapter 6 - Experiments

This chapter will contain the result of the experiments, challenges met and how they were overcome. This is aimed to be so detailed that it can be reproduced by the reader if that is desirable.

Chapter 7 - Discussion

The various mechanisms will have different effects on the baseline networks. We will analyse them separately and at the end try to look at them in respect to each other.

Chapter 8 - Conclusion

This chapter will be a short summary of our findings and what we concluded with regards to the research questions at hand.

Chapter 9 - Future work

Here we will list areas of the topic that need further investigation and problems we think should be looked into in future research.

2 Introduction to IPv6

In this chapter we will give the reader a short introduction to IPv6. The IPv6 protocol differs from its predecessor in multiple ways. In this chapter we will not go into great detail, but emphasise the biggest differences between IPv6 and IPv4. With this short intro we hope that the reader will be better capable to understand and follow the rest of the thesis.

2.1 Address notation

The first thing one will notice when looking at an IPv6 address for the first time is the longer and slightly different address length. IPv6 is using 128 bit addresses which are written in 8 blocks of 4 hexadecimal numbers each separated by colons. This in contrast to the 32 bit addresses used by IPv4 written as 4 blocks of decimal numbers separated by periods.

The IPv6 addresses are quite long in nature and remembering and notating the full length at all times would be unnecessary tedious and quite impractical. Because of this RFC4291 [20] defines some rules to shorten down the addresses for increased readability. These rules states that leading zeroes can be omitted, and *one* consecutive stream of zeroes can be denoted as a double colon (::), examples on the shortening rules are shown in Table 1.

IPv6 has also adopted the notation from IPv4 when notating ”subnets”. An IPv6 prefix is a block of unicast addresses and are specified with a backslash followed by a number that represent how many of the bits that are static. An IPv6 prefix of `2001:db8:1234:/48` tells us that the first, or left most, bits are static and set by the IPv6 provider. It also tells us that we have the rest of the prefix to do subnetting, a /48 prefix contains normally 2^{16} (65536) subnets each containing 2^{64} addresses.

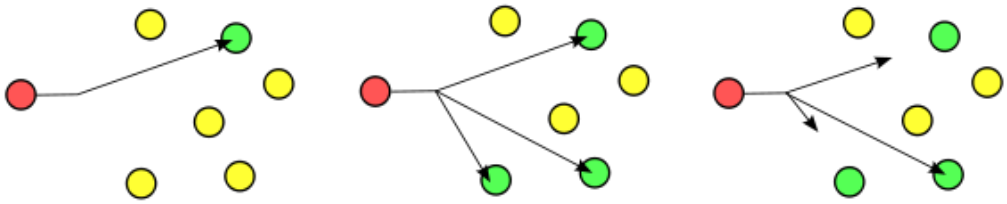
Table 1: Address examples

IPv4	192.0.2.1
Full IPv6$_1$	2001:0db8:0000:0000:0000:0000:0000:0001
Full IPv6$_2$	2001:0db8:1D00:0000:0000:0000:0000:0001
Full IPv6$_3$	2001:0db8:1D00:0007:0000:0000:0000:0001
Shortened IPv6$_1$	2001:db8::1
Shortened IPv6$_2$	2001:db8:1D00::1
Shortened IPv6$_3$	2001:db8:1D00:7::1

2.2 Address types

IPv6 addresses are grouped into multiple types of addresses based on the different ways the routers in the network are handling them. These types are called `unicast`, `multicast` and

`anycast`. A visualisation of the main types can be seen in Figure 1[1].



(a) Unicast (b) Multicast (c) Anycast

Figure 1: Address types

2.2.1 Unicast

The unicast addresses are the largest collection of addresses and are described by Hagen [2] in her book as, ”A unicast address uniquely identifies an interface of an IPv6 node. A packet sent to a unicast address is delivered to the interface identified by that address.”. There exists several special unicast addresses that have been reserved for special usage. We will enumerate these addresses and examples of the different types can be seen in Table 2.

Link-local addresses are used for ad-hoc networks and the hosts configures these addresses themselves. They have the prefix FE80::/10 and uses the interface ID as the last 64 bits with the bits in between zeroed out. These addresses are equal to the 169.254.0.0/16 addresses in IPv4 and are not routable.

Unique local unicast addresses replace the deprecated site-local addresses and as described in RFC4193 [24] they are intended to be globally unique but intended for local communication. They are not expected to be routable on the Internet but can be routed between multiple internal networks. It uses the prefix FC00::/7 but the RFC specifies that the 8th most significant bit has to be set to 1 for the time being and we get the prefix FD00::/8. The next 40 bits are proposed to be a global unique ID. the next 16 bits are subnet ID’s and the 64 bits are the interface ID.

Global unicast addresses is the name that in IPv4 are known as ”official” addresses. These addresses are world routable and are defined in RFC4291 [20] with the prefix 2000::/3. This address range are coordinated globally by the ”The Internet Assigned Numbers Authority” (IANA) which gives out address space to the organisations responsible for different geographical parts of the world, for an overview over the current allocations, see [25].

Table 2: Unicast variants

Name	**Prefix**	**IPv4 equivalent**
Link-local addresses	FE80::/10	169.254.0.0/16
Unique local unicast address	FD00::/8	N/A
Global unicast address	2000::/3	N/A

[1]All three pictures[21, 22, 23] used from the corresponding Wikipedia articles under the Creative Commons Attribution-ShareAlike 3.0 Unported License

2.2.2 Multicast

Multicast addresses are a collection of unicast addresses. This means that you can send traffic to one IPv6 address and by being a multicast address, every unicast address that are members of this multicast group will receive the traffic. Typical usage are streaming applications that sends traffic to multiple hosts, the job of giving all the participants access to the stream is then handled by the network itself.

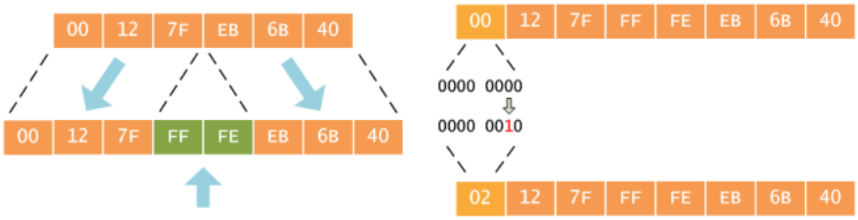
2.2.3 Anycast

In the same manner as multicast, the anycast addresses are groups of unicast addresses. The difference is that all hosts in an anycast group are offering the same service. A host is sending traffic to the anycast address and the network will send the traffic to the ”nearest” unicast address in the anycast group. This means that the host will establish contact with the closest service provider and get the best experience.

2.3 Autoconfiguration

To help network administrators when running IPv4 networks, a Dynamic Host Configuration Protocol [26] (DHCP) service is often deployed. This protocol gives the administrators the possibility to manage the addresses given out from one central position, and offer clients additional information of the network. This gives network administrators more work to do, and adds another point of failure. In RFC4862 [27] the authors describe a feature in IPv6 that is called stateless address autoconfiguration. Like the name says, this feature enables IPv6 hosts to automatically configure their own addresses.

An important part of this procedure is the generation of an interface ID. To generate this ID we use the 48 bit hardware address of the interface (MAC) as a starting point and reformat this to a 64 bit value that matches IEEE’s extended unique identifier (EUI-64) standard [28]. This is done in two steps that are illustrated in Figure 2[2]. The first step is going from the 48 bit starting point and to a 64 bit ID, this is done by inserting the hex value *FFFE* in the middle of the hardware address. When we have the 64 bit address we flip bit 7 of the address, this bit is by IEEE set to 0 in globally unique addresses, and by flipping this we state that this EUI-64 address is locally configured.



(a) Insertion (b) Flipping

Figure 2: Generating interface ID

[2]Both pictures [29] reproduced under the Creative Commons Attribution-ShareAlike 3.0 Unported License

As we have seen the default action for the client is to use the MAC address for the generation of the interface ID. This raises some privacy concerns as these MAC addresses are unique for every network interface card (NIC) that are produced and people are worried that they can be traced and identified on the Internet with this knowledge. The MAC addresses are also vendor specific and can make an attacker able to find machines with e.g. NIC device driver weaknesses. To cope with these issues an RFC titled ”Privacy Extensions for Stateless Address Autoconfiguration in IPv6” [30] was drafted, it addressed these concerns and have given us a way to create an interface ID at random. When these extensions are activated, the host will by itself at given time intervals change its IPv6 address to avoid always using the same address.

Every IPv6 host will create a link-local address with the aforementioned FE80::/10 prefix and the interface ID. To verify that this address has not been generated by another host on the network it is set to a *tentative* state, and a *Duplicate Address Detection (DAD)* process is started. This is done by sending traffic with the tentative address as the receiver. If no answer is received the address is unique on the network and the state of the address is set to *preferred* on the interface. This enables the host to communicate with other hosts on the same network as these addresses are not routable.

Like IPv4, IPv6 also has the option of giving the network administrators full control over the addresses being used by the hosts, DHCP has it’s own IPv6 variant called DHCPv6 [31]. This gives administrators that do not feel comfortable using autoconfiguration an option for setting up their address assignments themselves. DHCPv6 and stateless autoconfiguration are often used in parallel, autoconfiguration is used for address allocation and DHCPv6 is used to give the clients extra options on the network set up i.e. Domain Name Service (DNS) servers.

To create a global address, the host will send out a *Router Solicitation Message*(RS), these are multicast messages and will prompt a reply from a router with a *Router Advertisement Message*(RA). Every message has two flags to determine how the host should self configure, the ”O” and ”M” flag, Table 3 shows the different options. If no message is received in return, the host will try DHCPv6. But if such a message is received it will contain the prefix that should be used on the network and the host can then create a global unicast address by concatenating the prefix and the interface ID. Just like the link local addresses, the host goes through the DAD process to ensure that no duplicate addresses can exist on the same network.

Table 3: Autoconfiguration flags

”O”-flag	**”M”-flag**	**Description**
0	0	Only autoconfiguration available
0	1	DHCPv6 is used for address configuration but nothing else (unlikely)
1	0	Autoconfiguration is used for addresses and DHCPv6 for other configuration settings
1	1	DHCPv6 is used for all configuration

3 Related work

In this chapter we will try and sum up some of the previous work done in the field of IPv6 and the implementation in existing IPv4 networks. Section 3.1 will sum up what other researchers think should be the motivating factors for implementing IPv6 .Further on in section 3.2 we will have a look at what techniques that are proposed to help us in making the protocols coexist, and in section 3.3 we will sum up some experiences gained from existing systems running the two in parallel. The last part of this chapter, section 3.4, will sum up some security concerns that have been proposed with regards to IPv6 in this setting.

3.1 Why we should switch

For this project to have some purpose there must exist some incentive to make the switch from IPv4 to IPv6. The authors of the book "IPv6 Essentials" [2] points out that in the beginning of allocating IP-addresses there was little or no restrictions on the size of the blocks being handed out. This has lead us to the current situation where early adopters, and more specific American organisations and government controls approximately 60% of the allocatable IPv4 address space. This worked fine in the beginning, but the number of hosts on the Internet exploded, and IPv4 addresses is becoming an increasingly more sparse resource. IPv6 does not have this limited address space and will in the foreseeable future not have the same shortcomings on the number of addresses available. While the mere technical details of IPv6 solves some problems we are having with IPv4, the author calls the underlying new structure on the Internet and the extensions for the "...most essential advantage", in this context she is referring to the fact that IPv6 provides the foundation for a new generation of services. She especially makes a point of the ability to be ready for these new services and to be able deliver such services from an early stage as a key motivation for an early adoption.

3.1.1 Addresse space

The biggest challenge that IPv4 faces is the rapidly depleting available addresses free for allocation to new participants on the Internet. In Figure 3 made by the International Telecommunication Union (ITU) [32], we can see that the number of users on the Internet have exploded in the developed world in the time period 1997-2007, if the same growth is to happen in the developing world, we will see a massive shortage of addresses in the next year(s)[1].

In an announcement from January 2010[33], the Internet Corporation for Assigned Names and Numbers (ICANN), stated that we have under 10% of the total address space left for new allocations and the need to start adoption of IPv6 is essential. They concluded that:

> The Internet now defines communication and commerce and to accommodate its explosive worldwide growth we need to act now to guarantee an online future that accommodates growth with few limitations.

[1] At the time of writing the day of depletion is 31. July 2011 according to; `http://www.potaroo.net/tools/ipv4/index.html`

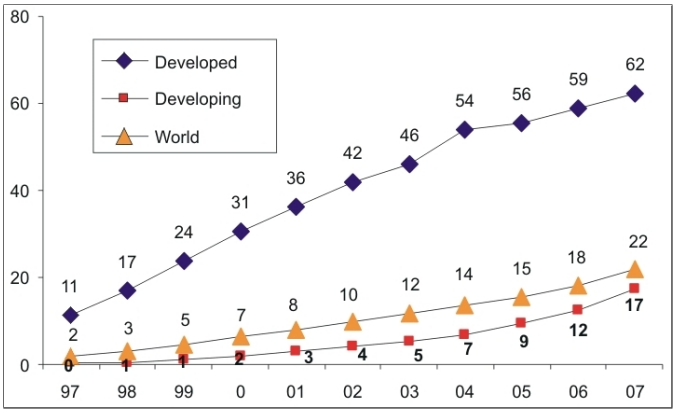


Figure 3: Internet users per 100 inhabitants 1997-2007

This means that if we do not do something to mitigate these exhaustion problems we will find ourselves in a position that makes new connections to the Internet impossible on the basis that we simply do not have any more addresses to allocate for new participants.

3.1.2 End to end connectivity

Ever since the very early beginning of the Internet scientists have described what they called the *end-to-end principle* with regards to connectivity on the network. This principle was born when Saltzer, Reed, and Clark [34, 35] started to discuss where not to put functionality in communication systems. They reached the conclusion that the end nodes should be the smart terminals and the network should be dumb in the sense that it just forwarded packets to its destination. In an early effort to standardise some ”Architectural Principles of the Internet” [36] this principle is given special attention, and the 8 years newer RFC3439 [37] keeps the exact wording from its predecessor.

> The network’s job is to transmit datagrams as efficiently and flexibly as possible. Everything else should be done at the fringes.

In the beginning of the Internet era this was no problem to live by. But very soon people started to realise that IP addresses was to loosely handed out and would eventually run out. This lead to the implementation of NAT. NAT enables many users to share the same official IP address, and thus giving the Internet community some headroom when it came to depleting the address space. Unfortunately the whole process utterly broke the *end-to-end principle* that once where the dominating way of thought. This has gone so far that an RFC was released with the title ”Unintended Consequences of NAT Deployments with Overlapping Address Space” [38] that describes scenarios that can arise when the usage of NAT is being exaggerated, and they reported that there exists ISP’s in developing countries actually deploying three level NAT internet accesses.

With IPv6 we have a golden opportunity to get back to the original idea of every host being able to address any arbitrary host on the Internet. The ISP will get prefixes that enables them to hand out global unicast addresses to every unit that needs them and can maintain end-to-end connectivity once again. When this connectivity is restored it will probably be a catalyst for new

innovative services on the Internet, peer-to-peer communication will be much easier without NAT traversal to worry about and could spawn protocols that enable us to use the network resources in a much more efficient way.

3.1.3 Routing

With IPv6 we get multiple benefits when switching from IPv4. Because of the aforementioned address shortage, the introduction of CIDR notation and widespread use of NAT exploded. This increases complexity in the way we use the available IPv4 address space. Parts of the existing IPv4 address space are reused around the world as a result of this shortage, with the result that we have routing tables without a clear hierarchical structure and that grows bigger in an increasing pace.

IPv6 incorporates what is known as hierarchical addressing structure. This means that no matter how big the organisation is, it can be possible to get a continuous address block and still just announcing 1 prefix to the Internet. Such an approach would make the routing tables on routers considerably smaller and thus faster. As an example on the difference is shown in Table 4 which shows the IPv4 routes Uninett[2] is currently announcing to the Internet compared to the IPv6 announcement.

Table 4: Uninett announcements

IPv4 announcements			
78.91.0.0/16	144.164.0.0/16	158.36.0.0/14	192.146.238.0/23
128.39.0.0/16	151.157.0.0/16	161.4.0.0/16	
129.177.0.0/16	152.94.0.0/16	193.156.0.0/15	*16 of them!*
129.240.0.0/15	156.116.0.0/16	192.111.33.0/24	
129.242.0.0/16	157.249.0.0/16	192.133.32.0/24	
IPv6 announcements			
2001:700::/32			*Nice single prefix*

To make this vision of a smaller hierarchical routing table come true, the *Regional Internet Registrars* (RIR) that are in charge of allocating IPv6 prefixes has detailed plans for the aggregation of addresses, RIPE-343 [39]. The *sparse allocation algorithm* make sure that they provide ISP's with room in the address space to grow bigger in the future, and thus trying to hold prefixes announced from one network down to one. The RIR's are also enforcing some strict conditions that have to be met before any addresses can be allocated [40], this to ensure that we do not get routing tables that grows uncontrolled. The conditions that have to be met are:

Be a Local Internet Registry (LIR) This reason is obvious, to become a LIR one must be member of one of the RIR's and accept their rules.

Don't be an end site Leaf nodes, or end-sites that do not provide connectivity to anybody else will not get their own prefix, they will have to contact an ISP to get a connection and a prefix from their allocated space.

[2]Uninett (ASN 224) is the Norwegian research network - `http://www.uninett.no`

Provide IPv6 connectivity by advertising aggregated prefix This is a threefold requirement. The first statement just says that you must plan to provide connectivity to external customers, assign them /48 prefixes (subnetable address space) and these prefixes must be advertised by the bigger prefix one will get. This last part is to make sure that ISP do not start to advertise individual routes for every /48 prefix.

Plan to assign 200 /48s in two years The last requirement is somewhat vague, it does not really demand anything but is sort of a threshold that one should strive to get past. The reason for this requirement is the new policy of keeping the routing table small and one can by this requirement get rid of the small businesses and private organizations that can do very well with a /64 assigned from an ISP.

The issues on how allocation of addresses to the end-users after the ISP/LIR has gotten the prefix is almost entirely left to the companies. However, IANA and the RIRs are not in any way interested in opening up for technologies like NAT that brakes connectivity again and have stated that the minimum size for allocation to end users are a /64 [40] prefix, giving every end site 2^{64} addresses and effectively mitigate the need for NAT.

Together with the smaller routing tables there are other advantages for the routers themselves. The header size of an IPv6 packet has been simplified quite much from IPv4, five fields have been removed and the size is now fixed at 40 bits. One of the fields that are removed are the *header checksum* meaning that the router no longer use processing power on calculating checksums on IP packets. Checksums are in most cases handled in higher layers and are thus redundant on the IP layer.

Fragmentation are no longer been taken care of by the routers. In IPv4 this is something that the routers would manage if they where to send a big packet over a network segment that could not handle it. IPv6 has moved this responsibility from the routers and back to the sender. Making the sender in charge of finding the *maximum transmission unit* it can send over the chosen path and thus moving even closer to the idea of the stupid network with all intelligence located at the fringes as previously described.

3.1.4 Security

IPv4 is an old standard, dating all the way back to 1981 [41]. When this RFC was crafted, the Internet of the time consisted of a set of well known hosts that where mutually trusted. Security considerations taken into account at the time were mostly about the robustness of the protocol making different implementations work with each other and ensure connectivity. This can not longer be said to apply to the hostile Internet we are seeing today, where unprotected hosts quickly will fall prey to attackers.

Murphy *et al.* [3] also points out that IPv4 was designed to be used in a network with a set of relatively trusted participants and have not any built in security features like authentication, confidentiality or non-repudiation in the protocol. It was not until 1995 with the add on IPsec [42, 43, 44] that IPv4 could offer layer 3 security in the protocol. Unfortunately IPsec are not mandatory and the feature is an add on that one can not take for granted when purchasing IPv4 equipment. Ever since the first release of IPv6 [1] IPsec has been a mandatory part of the

protocol, meaning that you can not label equipment IPv6 compatible if IPsec is not implemented, thus giving IPv6 well tested security options native in the protocol.

3.1.5 Controlling the unknown

Whenever new technology is being used on the Internet, people find new and obtrusive ways of using it. For system administrators this means that they will have protocols and technology running around in their network that they can not control.

Such traffic and usage can create ”unknown unknowns”, threats that you do not know you don’t know about. By educating people we can get them to realise that IPv6 could be such a threat and maybe get them to recognize it as a ”known unknown”, something we know about but are uncertain on the ramifications of.

When acknowledging the two previous statements we can then argue that the only way to get these threats down to the ”known knowns” is to incorporate systems that enables us to if no other then monitor such packets in our network. Networks that take the plunge and implement these technologies has then taken the choice of taking ownership over this technology and create policies for what is deemed as acceptable behaviour.

3.2 Transition tools

The ”Internet Engineering Task Force” (IETF) has long acknowledged the weaknesses that have emerged with IPv4 and did in 2008 publish a RFC called ”An Internet transition plan” [45]. This as the name implies is giving a detailed IPv6 implementation timeline that the authors think ISPs should follow to be ahead of the problems that will come with the address exhaustion. Unfortunately the RFC has not been widely adopted and it can be said to have been a somewhat failure in the aspect of getting the ISP’s to implement IPv6. The process described on the other hand looks very sane and could be used in the future with modified time goals.

Leng *et al.* [46] did in 2006 a study on the current status of the research done in the field of making the transition to IPv6. They concluded that much work had been done in the area but all would need more research. They especially mentioned scenario analysis, security analysis and how to make legacy IPv4 applications work in IPv6 only network as the biggest challenges.

Scenario analysis has been done in multiple cases, Bound [47] did in his paper define a set of 3 enterprise network scenarios that he developed a set of pertinent questions for. This to allow enterprise administrators to refine their deployment scenarios and thus making them get all the aspects into the light before advancing to actually implementing IPv6.

The authors of RFC4852 ”IPv6 Enterprise Network Analysis - IP Layer 3 Focus” [11] have done some more scenario analysis and focuses on the larger networks that are characterised by multiple internal links and dedicated router connections to one or more service providers. They list ten different transition scenarios that they assume will be encountered when doing the implementation, they continue to analyse and recommend techniques for each of the scenarios. Network administrators that have taken the step and decided to implement IPv6 in such networks get good help in designing and avoid the big security issues with this list.

In their book Hogg *et al.* [4] defines the transition mechanisms as a threefold. The *dual-stack solution*, all the nodes have two IP stacks and both are in use simultaneously utilising IPv6 to

contact IPv6 hosts and vice versa. *Tunnel techniques*, various tunneling modes are used to encapsulate IPv6 packets into other protocols, the *translation methods*, acting as a gateway between the IPv6 and IPv4 world. All together in this threefold there exists quite a lot of techniques which all have its pros and cons, some are no longer relevant and some need further analysis before we can say that our network can benefit from them or not. The techniques found most relevant are examined further in Chapter 4.

3.2.1 Dual stack

Dual stacking is in many publications referred to as a transition mechanism [2, 3, 4], this is somewhat not quite correct. The technique is not about implementing some mechanism that helps IPv6 to function in a special way. Dual stacking is about rolling out two separate independent networks and make all hosts available on both. According to the authors this is a preferred way of doing the implementation, but there exists some special requirements that has to be fulfilled. One must have an IPv6 connection to the Internet and have a network that can support native IPv6 on its infrastructure.

While dual stacking does help us in the transition, it is of no help if our networking equipment does not support IPv6. Sailan *et al.* [48] pointed out the need for policies that dictates the purchase of IPv6 ready equipment and mentioned the ”IPv6 Ready Logo Program” [49] as a source for network technicians to verify IPv6 compatibility before making the purchase. This can be the first step in a long term plan of implementing IPv6 in the network, making sure that new purchases is supporting IPv6 when replacing old equipment and thus readying the network for dual stacking over time.

3.2.2 Tunneling

RFC 4213 ”Basic Transition Mechanisms for IPv6 Hosts and Routers” [50] and RFC 3056 ”Connection of IPv6 Domains via IPv4 Clouds” [51] makes the assumption that the network only has access to an IPv4 Internet connection. Their proposal of configured tunneling require a router/host that can terminate a tunneling connection available somewhere in the cloud. This will enable the network to get an entry point for IPv6 that will to the router look as a native connection. These solutions does however require a company delivering such services and will because of this in many cases raise cost issues.

Carpenter *et al.* proposed a mechanism called 6to4 [51] that addresses this issue with the need for a business partner in the cloud. They based their tunneling on much of the same principles but got a anycast address on the Internet to handle the traffic, this means that the service is offered as a community service and volunteering ISP’s can cooperate in making it function properly by adding routers that handle traffic to the anycast address.

Further research into 6to4 tunneling is done by Savola *et al.* [10]. They expressed the concern that all tunneling techniques are built with the notion that central nodes have to accept all traffic for it to work. This means that 6to4 nodes are not able to identify if a node is legitimate or an attacker, this is the Achilles heal of the technology. The authors do not have an answer but acknowledges that the protocol has got some required/implied security checks when doing the encapsulation/decapsulation that will mitigate much of the security risks if implemented properly.

The Intra-Site Automatic Tunneling Addressing Protocol (ISATAP) [52] is another mechanism that bases its functionality on tunneling, it differs from 6to4 in the sense that it is not an automatic tunneling mechanism. To function it has to have a special host called an *ISATAP router*, this host is responsible to give out the IPv6 connectivity to the hosts, these hosts also need a special client to function properly. This technique is implemented in many different operating systems and Visoottiviseth *et al.* [53] have in their work from 2008, looked at the performance difference between the implementations. They conclude that the network performance is suffering from the encapsulation, but the performance drop is very low until the traffic is quite substantial. With this they add that administrators may base their routing platform based on management experiences and ignore the performance differences.

Teredo [54] is a tunneling protocol that is aiming to give IPv6 access in networks that employ NAT devices via UDP tunneling. Huang *et al.* [55] have reviewed the protocol and analysed the detailed workings before they implemented it and ran performance tests on the added latency from the technique. They could report an average added delay of 9.03µs with their implementation when traversing IPv6 traffic directly through the NAT device.

Hoagland [9] takes a close look at the Teredo protocol from a security point of view. The author emphasises the fact that the paper is based on the protocol specification alone and not current implementations. They stress the fact that Teredo is putting the end client directly onto the IPv6 Internet and the need for security applied after the tunnel is terminated. This will in a practical point of view be on the client itself thus bypassing security in the network making the need for a host firewall paramount. They do however recognise the work gone into the protocol in terms of sanity checks on the different components that will effectively prevent many potential attacks.

3.2.3 Translation

The ”Network Address Translation/Protocol Translation” (NAT-PT) [56, 57] technique is studied by Vazão *et al.* [58] as a mechanism to secure the interoperability between hosts on the different IP stacks. They implemented the NAT-PT in C for operating systems using the Linux kernel in their experiments. From their experiments they concluded that compared to other Dual Stack, and tunneling options, NAT-PT shows the most promising result. Waddington *et al.* [59] has in their article also looked at different techniques to ease the transition and concurs with the previous authors that the mechanisms that are best suited at this stage is NAT-PT, but also sees the tunnel technique 6to4 as a useful supplement. Although this technique had gotten much praise from academia, Aoun *et al.* [60] in 2007 summed up papers that described shortcomings of the protocol and got it approved by the IETF, effectively moving NAT-PT to historical status and are not longer a recommended protocol.

The previous mentioned dual stacking will function as long as the network has got enough IPv6 and IPv4 addresses to all their clients. One possible scenario that Xia *et al.* [61] have looked at in their paper, is what we are going to do when we have more hosts on the network than IPv4 addresses available. The solution they have looked at is called DSTM [62], this works by keeping a pool of IPv4 addresses that the clients will get assigned for a short period of time when they need to contact an IPv4 host. When the communication is finished, the address is made available

for other clients, and will with this be able to support multiple IPv6 hosts with a scarce number of IPv4 addresses. The mechanism is described as light weight, but they emphasise the need for client software on every host and the small impact such mechanisms will have in the network community before IPv6 has gotten a stronger hold in the market. It is however something that also needs research as the scenario is highly possible.

An approach that combined the two previous translation mechanisms was proposed by Kong *et al.* [63]. The technique was based on a translator in the cloud that used the dynamic host addressing scheme from DSTM to retrieve an IPv4 address from the ISP when a connection request came in. The translator could then talk directly to the IPv4 only device while using the translation technique from NAT-PT to give the traffic back to the IPv6 only host requesting the connection. The authors implemented and tested the proposed system with good results. In retrospect, the techniques that is meant to work in predominantly IPv6 networks is not getting much attention from the networking community. This is something that has to change in the future when these configuration become more and more common.

3.2.4 Comparison

Comparison of the different techniques have been made by Waddington [7] in 2002 and a similar paper by Govil *et al.* [8], in 2008. The first paper compared 13 different mechanisms and was concluding that NAT-PT, 6to4 and configured tunnels were they way to go. When much of the same work was done over again in 2008, the techniques to compare had risen to over 20 transition techniques. They examined them all briefly but were unsuccessful to recommend some over the others and concluded that there are no single best solution for the task. Site specific constraints should be taken into consideration and they do not recommend any technique over the other at the time of writing the article.

3.3 Experience and performance

Auburn university did in 2002 start what we now can say was an early adoption of IPv6. They choose two subnets that were especially meant for testing and two that were in daily use. They used a tunnel to set up the connection to the world and started to test. They ran into connectivity issues and further on problems with DNS services but got them sorted out. These problems was not an issue with IPv6 itself, but unforeseen complications in their existing network. In their report, Carlisle *et al.* [12] drew the conclusion that the upgrade to IPv6 was only a matter of granting enough money, training personnel and freeing up time for the right personnel to do the job.

Yan-ge *et al.* [13] have published the details on how the Xuchang university set up the implementation of IPv6. They have deliberately split their implementation in two different segments. One segment that ran IPv6 only and used a NAT-PT gateway that handles the routing. The second segment were a dual stack environment. Both techniques where found to work fine side by side and has made the authors able to deploy IPv6 on the network as a whole and getting both types of environments functional for further research.

Performance measurements on network hosts running dual stack configuration have been tested by Law *et al.* [64] and Wang *et al.* [65]. Both authors have tested on a large number of

hosts, 2014 and 936 respectively, but with 3 years difference in time. Wang found in 2005 the global IPv6 performance to be in need for more connectivity, reducing the packet loss and stabilize the Round-Trip Time (RTT). They also made the observation that the tunneled connections did not suffer notable performance degradation compared to the native IPv4 traffic. 3 years later in 2008, Law did much of the same experiment and observed that the connectivity of the IPv6 backbone was now 95%, meaning any IPv6 host could reach 95% of the online IPv6 hosts. The RTT was still high, due to longer propagation distances, but the throughput of the IPv6 backbone because of its light usage was superior to IPv4. They also concurred with Wang on the fact that tunneling mechanisms had little or no impact on the overall network performance.

DNS servers are one of the biggest contributors to the functionality of dual protocol networks. They work on a higher level than IP and do not care if the requests comes in over IPv4 or IPv6. They concern themselves with handling the requests and giving out the requested information, A records for IPv4 hosts and AAAA records for IPv6 hosts. Hiromi *et al.* [66] have in their paper described problems with DNS and IPv6, the authors found DNS servers that had the correct AAAA records for IPv6 hosts but ignored it and returned the A record for the same request. This is not a problem that is a direct consequence of the IPv6 protocol itself, but is an important part of making deployments work and should be handled accordingly.

3.4 Security considerations

When looking at the security considerations one have to address when doing the implementation of IPv6, it can seem like a daunting task. Davies *et al.* [67] have chosen to divide the security problems into three main segments.

Issues due to the IPv6 protocol itself The authors goes through a plethora of problems regarding the protocol itself, routing headers, Multi/Anycast, ICMPv6, fragmentation and more exotic issues. Most of these issues are manageable and are can be viewed as a list of problems that should be taken into consideration when deploying.

Issues due to the transition mechanisms The coexistence of transition mechanisms in complex networks will by their believe increase the chance of vulnerabilities in the mechanisms themselves, in the interaction between them or by introducing paths through existing security services.

Issues due to IPv6 deployment The third part of the threefold are focusing on the fact that IPv6 is new to many administrators and have a multitude of pitfalls that can be made. The authors make a point of not taking on IPv6 lightly, ”just doing a small pilot”, it should be handled with care and thoroughness so that problems with bad design choices does not affect the network.

Another paper that are doing much of the same as the previous author is Lancaster [68]. He has split up and analysed the problem of implementing IPv6 in much the same fashion and has summarised his recommendations into a list based on when actions should be taken.

Immediate Actions to take before IPv6 Deployment When the administrators become aware of IPv6 they should take some actions before they start to roll it out in their organisation. Prevent unknown IPv6 attacks by filtering known tunneling techniques, research the possible adap-

tation of the IPv4 firewall to IPv6 and get to grips with the IPv6 protocol to be able to spot abnormal activities.

Immediate Actions to take on IPv6 Deployment Just after the deployment of IPv6 they should start to test their firewall over IPv6 and testing internal hosts with special attention to IPv6. The author also stresses the importance of keeping the policies ”alive” and evolve them over time.

Preparation for the Future To be able to keep up with possible attackers, administrators must keep up with current and future technologies. They should, according to this paper, already now look at de-centralized firewalls as this is something that is coming in the future.

The latter of preparing for the future and looking at distributed firewalls has been done by Lai*et al.* [69], they have designed and implemented a distributed firewall system meant to be used in IPv6 networks. The term distributed firewall system, is here understood as the combination of a plain perimeter firewall located on the edge of the network filtering all Internet traffic, a host based firewall situated on every host in the network and a policy centre that controls the rules executed by the firewalls. They address the need for the system to recognise IPv6 traffic in all its forms, this includes native, tunneled or in other ways obfuscated packets. They conclude with the notion that all though this is a prototype and should be developed further, it looks promising and would yield a more secure network.

Caicedo *et al.* [6] described in their paper a set of security issues regarding IPv6 deployment, they concluded that there are security issues with doing the transition, but with the right implementation and especially with a good system for IPSec and PKI infrastructure, on could mitigate the biggest threats. They also made the point that we do not yes fully understand the cost implications and do not have the tools available for proper planning of the transition.

4 Transition mechanisms

To help us operate our networks while going from IPv4 to IPv6 we have an extensive list of transitional techniques that in different ways promises to help us. These techniques are separated into three distinct groups, Section 4.1 will look at dual stack configurations. Section 4.2 will look at the tunneling techniques, and in Section 4.3, transitional protocols are the target. We will give the reader a clear understanding of the operational characteristics of the techniques, investigate on the pros/cons. After we in Section 4.4 give a summary of the transition techniques we have covered and chosen the ones that will be used for further analysis and experimentation, we will have answered the first research question posed in Section 1.5.

4.1 Dual stack

As the name implies, this technique is to use an operating system that have the possibility to use both IPv6 and IPv4 on the host. This means that the host retain a complete IPv4 stack but also have added the IPv6 stack in addition. This leaves the host with the option to function in three distinct modes.

IPv4 only The machine has both stacks installed but the IPv6 part is disabled. Used in IPv4 only networks.

IPv6 only The machine has both stacks installed but the IPv4 part is disabled. Used in IPv6 only networks.

IPv4/IPv6 Both stacks are active at the same time and the operating system switches between them depending on the hosts it is communicating with.

These two stacks are independent of each other and are configured completely separated, IPv4 has it's own mechanisms for configuration, normally DHCP, and IPv6 can rely on autoconfiguration or DHCPv6 or they are simply manually configured.

To choose which IP-version dual-stack hosts should use they first check that they are able to configure their IPv6 interface correctly, and they use DNS requests to get additional information on the host they want to contact. If the host is IPv4-only, they will get a single A-record, but if the host is dual-stacked or IPv6 only, they will most likely get two records, an A and AAAA (quad A) record. At this point the host will have two equally valid routes to the destination and it is all up to what IP-version that have been configured as the preferred one. In these situations IPv6 is preferred by default in all the major operating systems today, Windows 7, Mac OS X and multiple Linux/Unix distributions [70, 71]. The process can be described in 4 simple steps:

1. The host tries to configure the different IP-versions, and enables the versions that are successfully configured, normally done at boot up.

2. It sends out a DNS-query to its primary DNS-server for the wanted domain, e.g. smtp.example.com
3. The DNS-server replies with the corresponding records, A-record for IPv4 (`192.0.2.25`) and AAAA-record for IPv6 (`2001:db8::25`).
4. The host receives both of the records and uses the preferred record type to contact the host.

An infrastructure that treats both versions as independent systems, meaning all network equipment has both stacks and are configured separately, are called a *dual-stack network*. In such networks there are some extra costs with supporting two different communication protocols at the same time. According to Hogg *et al.* [4] and Frankel *et al.* [72] such costs will be:

- Sharing of network bandwidth between the protocols
- Increased router load from:
  - keeping, maintaining and executing forwarding tables for both protocols
  - running routing protocols for IPv4 and IPv6
  - doing packet filtering for both protocols
  - handling the different special cases for both protocols
- The hosts must use more CPU, memory, network traffic etc. to run both protocols.
- The administrative staff has to learn the new technology and use time to plan and execute the implementation, running concurrent protocols also makes the networking environment more complex and adds work load to the network administrators.

The biggest problem in IPv4 today is the shortage of available addresses, this means that a dual-stack host will not help us mitigate this problem if it needs one official IPv4 address each to function properly. This means that we need mechanisms that allows us to build IPv6 only internal networks with some added functionality on the network edge to allow for communication with IPv4-only hosts, or mechanisms that allows the hosts to be world routable on IPv6 while IPv4 can rely on private addresses. Such mechanisms will be discussed in further detail in later sections.

4.1.1 Security considerations

When we are talking about security in dual-stacked environments we have to keep in mind that every major operating system from around 2006 (Windows Vista and onwards) ships with activated and preferred IPv6 stack. This means that we can effectively differ between two types of dual stacked networks. Intentional dual stacked networks, where the network administrators have made a decision to allow IPv6, and unintentional dual stacked networks, where IPv6 enabled hosts are locked down in IPv4 but are IPv6 are left uncontrolled.

Organisations that have made the decision to implement IPv6, and by this getting an intentional dual stacked network, will need to look out for some key points that need to be addressed when they are planning the implementation.

Security policies One thing that security officers in organisations must be clear about is the need for thought through security policies for IPv6. When implementing IPv6 you effectively

double the amount of ”roads” into your network. In fact, existing transition technologies will not only double the ”roads” but add a way in for every host if not properly managed. The same strict and well defined policies should exist for IPv6 as the existing ones are for IPv4. This means that every policy should be updated to take into measure IPv6 as an factor when designing them.

IPv6 functionality In its nature, IPv6 brings some new functionality to the network. Autoconfiguration, neighbour discovery, end to end encryption and unexpected tunneling are all things that needs to be addressed. All of which act very different from IPv4. The aforementioned security policies must take all this new functionality and develop policies for the network so that they behave in a predictable fashion.

Existing systems Making sure that all existing systems can handle IPv6 are paramount. Getting an inventory over the existing equipment and compile a list over where the problems may arise. The firewall must be able to handle IPv6 traffic and develop packet filters that enforce the policies for the network, intrusion detection systems must be ready to audit the new packets, and the administrators must make sure that they actually works.

Preferred IPv6 When the decision is made to use IPv6, there should also be a clear policy that dictates the preferred usage of protocols. IPv6 should be the protocol that are default and IPv4 should be faded into the background. When hosts no longer need IPv4, it should be disabled in an effort try and minimise the usage of dual protocols. The added complexity you get when running two protocols at the same time should be a priority to get rid of.

In networks that are by policy said to be IPv4 only, network administrators can suddenly find that they have an unintentional dual stacked network. This is something that should be taken seriously, and steps should be taken to disable unwanted dual stacked hosts. Some hosts in the core network should in such a case be IPv6 ready, audit for router solicitation messages, network solicitation messages and other signs of IPv6. When such hosts are detected an alert should be made so that administrator can take the necessary steps to disable the unwanted broadcasting. If this is not done we could find ourselves in the position that we have rogue machines in our network sending out RAs that effectively route all IPv6 traffic through themselves and listen in on all the traffic.

4.2 Tunneling

With the previously mentioned dual-stack approach to the problem, we make one important underlying assumption, namely the assumption that the network provider is able to give us a direct connection to the Internet via IPv6. This is in many cases just not an option when very few ISP’s offer native IPV6 connectivity to their customers, and is naturally a problem when companies want to add IPv6 functionality to their networks.

One solution to this problem can be what is called *tunneling* techniques. Tunneling is also widely known as *encapsulation* and these two are often used interchangeably. Both of them describes the technique of using one protocol (e.g. IPv4) to transmit another protocol (e.g. IPv6), this means that we use the existing IPv4 infrastructure to move our IPv6 traffic. This encapsu-

lation is illustrated in Figure 4. To make this work we need dual-stack nodes that can talk both protocols and be able to execute the three basic components of the process:

Encapsulation The originating network must have a defined node that takes the original protocol packets, encapsulate it into transport packets and sends them to a receiving node that are aware of the encapsulation.

Decapsulation The packet from the aforementioned node must have a counterpart that can take the transport packets, strip away the transport header and relay the original protocol to the appropriate network.

Tunnel administration Both the sender and receiver must agree on how they should operate such a tunnel so that the packet flow not will be interrupted.

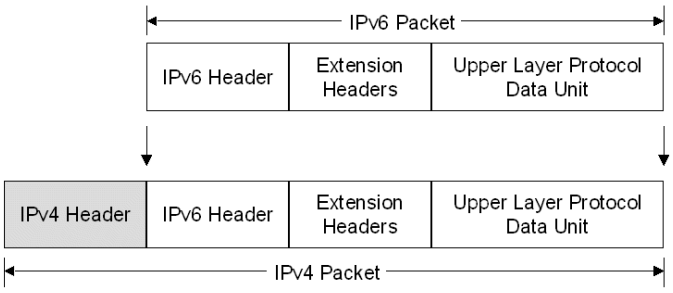


Figure 4: IPv6 packets encapsulated in IPv4

One important aspect of the tunneling is the use of protocol 41. Every IPv4 packet has got a header that precedes the actual data we want to transfer, this is a set of bits used for giving the receiver information about the packet that is arriving. In the header we set the *protocol* field to the decimal value of 41 when we are using encapsulation. This value is acknowledged by IANA [73] as the default way to recognise encapsulated packets and the receiver will by this recognise and decapsulate the IPv4 packets before forwarding the IPv6 packets to the final destination.

When we are talking about tunneling, we divide them into two different types based on their functionality. Configured tunnels are static tunnels that are configured manually and automatic tunnels that the hosts/network initiates and uses automatically.

4.2.1 Obsoleted tunneling mechanisms

The following two tunneling mechanisms are obsoleted or just not getting any momentum in the battle between the different tunneling techniques. We mentioned them here as they have ideas and techniques that have outlived their original proposals and gives us some understanding of the maturing process involved when creating newer mechanisms.

Automatic tunneling

In RFC2893 ”Transition Mechanisms for IPv6 Hosts and Routers” [74] the notion of automatic tunneling was first introduced. It was based on an assumption that the end host could play

a bigger role in the creation of tunnels to the IPv6 hosts they wanted to contact. This means sending a packet with the `::/96` prefix, which is set aside for this purpose, the 32 right most bit of the address being the IPv4 address. This was then set as the sender address and passed on to hosts that would read the IPv4 address and encapsulate the IPv6 packet in an appropriate IPv4 packet, forwarding over IPv4 to the corresponding host that would receive and decapsulate the packet. With this idea in mind one can wonder why the two hosts not simply talk IPv4 to each other. The proposal in this RFC was later revised and replaced with RFC4213 that removed this version of automatic tunneling. The idea however is still in effect, and multiple techniques are proposed on the same notion that the end hosts should be able to set up the tunnels it need for itself.

6over4

Like the RFC2470, ”Transmission of IPv6 Packets over Token Ring Networks” [75] uses Token Ring as the layer 2 protocol, and and RFC2464 ”Transmission of IPv6 Packets over Ethernet Networks” [76] uses Ethernet, we have a transition mechanism that can be said to use IPv4 as the layer 2 protocol, RFC2529 ”Transmission of IPv6 over IPv4 Domains without Explicit Tunnels” [77]. By this we mean that the existing IPv4 infrastructure is acting as the LAN.

In Section 2.3 we saw that IPv6 uses multicast when the node does the the autoconfiguration, this means that multicast also must be available on the underlying IPv4 network for 6over4 to work. When this is the case, the 6over4 mechanism enables the IPv6 host to do full neighbour discoveries on top of IPv4 and thus making IPv4 in reality a layer 2 protocol similar to the aforementioned Ethernet and Token Ring.

The problem with this approach is that multicast in IPv4 networks is an add-on to the IPv4 protocol[1] and is not a mandatory functionality in IPv4 deployments. 6over4 is also lacking implementation from the biggest operating systems and by having to rely on functionality that are in many cases lacking, the 6over4 transition mechanism is losing against other transition mechanisms that do not have these requirements. However, the notion that one could use the underlying IPv4 network as the layer 2 protocol, have been ”stolen” from 6over4 to newer more used transition mechanisms.

4.2.2 Configured tunnels

Configured tunnels can be described as static tunnels that network administrators set up and maintain manually. These connections are established before any traffic arrives and often acts as an invisible bridge to give the network IPv6 capabilities. This makes the tunnels work on a *router-to-router* basis and gives the network administrators the opportunity to get a fully working IPv6 connection via their existing IPv4 only Internet Service Provider (ISP). There are two scenarios that most likely will involve such tunnels:

Offered by ISP The case where your ISP have some IPv6 capabilities in the core network, but can not offer the IPv6 all the way to the end customer. In this case one could set up a router in the ISP’s core network that could terminate a tunnel and give the customers the opportunity to get IPv6 over the IPv4 link already in place, enabling ISP’s to buy some more time to

[1] In fact IPv4 multicast is almost as ”young” as IPv6.

implement full scale support.

Tunnel broker The cases where your ISP does not have any IPv6 capabilities. There exists multiple tunnel brokers on the Internet[78, 79], these brokers offers IPv6 in IPv4 tunnels with points of presences (POP's) in geographically different places and thus making it possible to get a tunnel with low latency for the IPv6 connection. Tunnel brokers are described in RFC3056 "IPv6 Tunnel Broker" [51]

Configured tunnels are predominately a way of getting an IPv6 connection that is transparent for the overlying users and hosts. This means that for the internal dual stacked host in Figure 5 the encapsulation/decapsulation process that the border router is doing when routing IPv6 packets is invisible, and the IPv6 connection is seen as native for the internal clients.

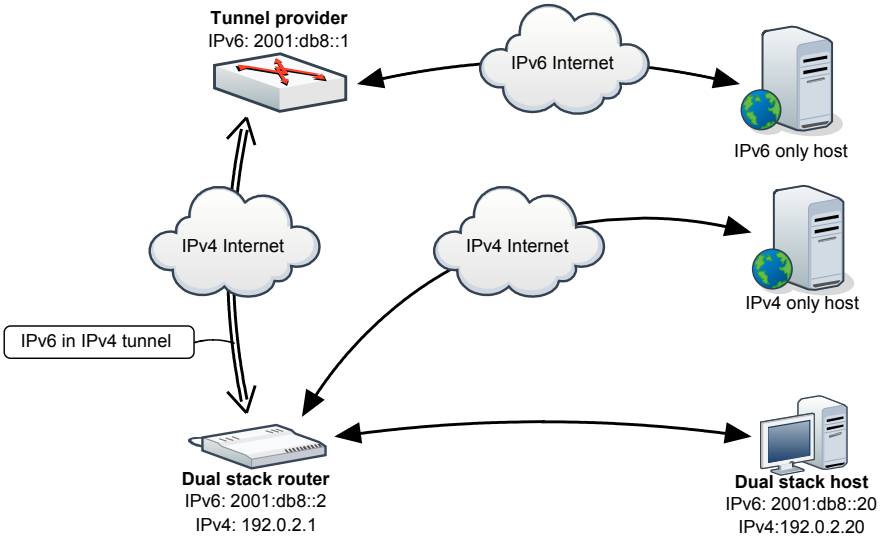


Figure 5: Configured tunnel

Security considerations

Tunnel endpoints that exists in the IPv4 Internet and are allowing us to set up the tunnel are in most cases beyond our control. By the operational characteristics of such encapsulation/decapsulation points required to accept all traffic from the IPv6 Internet that want to communicate with the prefix we are assigned. When we look at security issues with configured tunneling we should make the assumption that the traffic entering on tunnel endpoints are not to be trusted, they should be considered an external link. With this assumption clearly stated we can then differ between the two end points in these tunnels. One is the tunnel endpoint at our tunnel provider. The other is the router terminating the tunnel on our end. At the provider end there not many things that should be done, we want all the traffic that are coming our way so that we do not risk wanted traffic to be dropped before it reaches our own security perimeter.

The traffic that arrives at our dual stacked border router can be viewed as two different packet streams. The first one being the IPv4 traffic with the encapsulated packets, and the second one

being the decapsulated IPv6 packets. This means that we should deploy one set of security checks on the encapsulated packets, filter out the unwanted ones, and only decapsulate the ones that meet our policy requirements.

With the encapsulated traffic we have the advantage that we have one specific host we know it should come from, the tunnel endpoint. Packets with IPv4 source addresses not from the tunnel endpoint but with tunnel headers should be dropped. We are not interested in traffic unknown senders have encapsulated. Of course source addresses are easy enough to spoof and additional security should be in place on the tunnel. The easiest way of doing this is to implement IPSec or other cryptographic mechanisms on the IPv4 connection between ourselves and the tunnel endpoint. If implemented correctly this will ensure that the integrity of the packets are kept and we can be sure to only accept tunnel traffic from our legitimate tunnel provider. This will not protect us from malicious packets sent to us from legitimate IPv6 hosts on the Internet, but ensure that no 3. party can send rogue traffic directly to our tunnel interface.

Only when the above checks have ensured that the packet is really a packet from our tunnel provider do we decapsulate it. After the decapsulation process we have the aforementioned second packet stream, an untrusted IPv6 stream that must be considered hostile and sent to the firewall for further scrutiny. At this point the tunneling mechanism has done its part and made sure that IPv6 packets do not enter into the network from other sources than the IPv6 tunnel endpoint.

4.2.3 6to4

Just as the obsoleted mechanism, automatic tunneling, the 6to4 mechanism[51] uses the existing 32 bit IPv4 address to help generate the IPv6 address it uses. The mechanism can be used by single hosts or by a complete network. Common for them both is that the network/host need to use a global IPv4 address. This means that hosts with a RFC1918[80] address, typically hosts behind NAT devices, can not use 6to4 by themselves. For 6to4 to work in these configurations, the host with the external IP needs to run the IPv6 traffic for the entire network.

The features we have just described comes from the fact that IANA has reserved the `2002::/16` prefix for usage with the 6to4 mechanism, and it uses this to construct the IPv6 address. The host puts the 32 bits from the IPv4 address after the 16 bit prefix and thus making a `/48` network available for itself. This means that every officially assigned IPv4 address with 6to4 can create their own `/48` prefix network.

As we can see in Figure 6 there exists two different types of hosts that make 6to4 work, the *6to4 router* and the *6to4 relay*. These two have different scopes and differ quite a lot in their functionality.

6to4 router The 6to4 router sits on the edge of a network that it wishes to do the 6to4 mechanism for. It has to have at least one globally routable IPv4 address to make the 6to4 prefix and with this it will handle the routing for the entire network. The internal hosts will use normal IPv6 mechanisms for configuration and does not view the 6to4 router as anything other than the gateway router that handles IPv6. The 6to4 router can also be an internal component in every machine provided that every machine has an official IPv4 address. This can be suitable

for some usage but would create much overhead when talking to internal hosts as all hosts would have to set up IPv6 in IPv4 tunnels when communicating internally.

6to4 relay On the Internet there exists a set of dual stacked hosts that advertise routes to the `2002::/16` network taking all the traffic to this prefix from the nearest IPv6 node. These special hosts are assigned the global IPv4 anycast address `192.88.99.1` and encapsulated 6to4 traffic that are sent to this address are decapsulated and forwarded onto the IPv6 Internet. These 6to4 relays are the core of the mechanisms and enables the traffic to flow to and from 6to4 hosts.

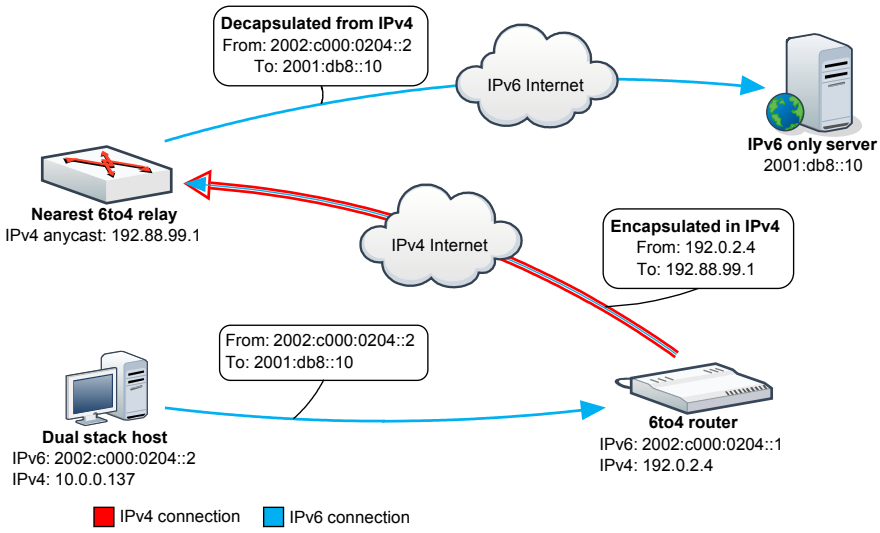


Figure 6: 6to4 network flow

For a network with a border router that have assigned the global address `192.0.2.4`, can construct the IPv6 prefix of `2002:c000:0204::/48` and use all the subsequent addresses. By this the network have a plethora of addresses to its disposal and can be configured in a multitude of ways. This however requires that the global address is static, or the IPv6 prefix will change with the IPv4 address. The host that constructed the 6to4 IPv6 prefix and have got the official IPv4 address is by this fact the default route for the IPv6 prefix and must be able to handle incoming and outgoing connections to and from the IPv6 Internet.

After the network is configured and is up and running with working IPv6 hosts we need to get packets to and from the IPv6 Internet. If no known relay routers are configured, all outgoing IPv6 packets should be encapsulated in IPv4 with protocol 41 and sent to the anycast address. When they get these protocol 41 packets, they decapsulates the IPv6 packet and send them onto the IPv6 Internet. When they get an IPv6 packet addressed to a `2002::/16` prefix, it takes the embedded IPv4 address from the IPv6 destination address and encapsulates it with protocol 41 into a new IPv4 packet with the embedded address as the destination. This packet is then forwarded over the IPv4 Internet to the corresponding 6to4 router/host.

One problem however with 6to4 is the absolute need for an global unicast IPv4 address to be able function at all. In countries where the availability of these addresses are sparse or the ISP want to save money, we see the usage of NAT on the ISP level. This means that every customer will get a RFC1918 address and are thus not able to use 6to4 to access the IPv6 Internet.

Security considerations

When we look at Figure 6, we can see that because of the usage of IPv4 anycast for the discovery of the nearest 6to4 relay the technique have some interesting characteristics security wise. Both the relay and border router has to unconditionally accept traffic from anyone claiming to be a 6to4 router.

The border router has to accept traffic form every host that claims to be a 6to4 relay as the 6to4 relay used for the return traffic is chosen by the sender at the time the packet is being sent. Although the border router has to accept all encapsulated packets it only functions to the outside world on IPv4 and needs only to do the ingress filtering on these packets. This means that it should drop packets that have destination addresses to other prefixes than their own and clearly erroneous source addresses. In IPv4 such address are; private, loopback, unspecified, multicast, broadcast, DHCP link local, or a reserved addresses. The border router is also connected to a well known internal network with both IPv4 and IPv6. This gives it the opportunity to apply egress filtering to avoid erroneous or malicious packets from leaving the network.

Like the border router, the relay router has to forward traffic that is being sent to it. Based on the fact that all relays have to be dual stacked, they all have one extra attack vector compared to the border router. The IPv6 Internet. It can protect against obviously spoofed packets, packets that have plainly wrong source addresses. From the IPv4 side it should do the same ingress filtering as the border router, and from the IPv6 side it should drop IPv6 packets that have source addresses that are IPv4 compatible, IPv4 mapped, loopback, unspecified, link local, site local, or multicast.

Despite the application of ingress and egress filtering, the 6to4 hosts are still vulnerable to other types of packet manipulations. They are well suited to be used in denial-of-service (DoS) attacks as they forward any packet with an approved IP address. This means that attackers can use such relays to do reflection attacks on hosts they would like to DoS, spoofing the source address to the victim and send to a large number of hosts which all reply to the same source address. This method will also ”launder” the packets as they perceived sent from the 6to4 relay and not the real sender.

4.2.4 Intra-site automatic tunneling addressing protocol (ISATAP)

Like the 6over4 protocol, the aim for ISATAP is to use the IPv4 infrastructure as a link layer for the IPv6 implementation, but the big difference from 6over4 is that it does not require the IPv4 network to have multicast enabled. As we have seen in Section 2.3, the IPv6 client uses multicast to determine information from the network that it needs to function, so ISATAP has addressed the need for IPv6 multicast on an IPv4 network that does not support it.

The replacement for the router solicitation multicast packet is the *Potential Router List* (PRL), this list contains IPv4 hosts that accepts IPv6 router solicitation messages encapsulated in IPv4 and can give the ISATAP host the information it needs to configure the IPv6 interface. The most

common way of generating these PRL's is to use the IPv4 networks DNS capabilities or get it with extra DHCPv4 options, a DNS query to *isatap.example.com* will in many cases be the preferred location technique.

Like normal IPv6 autoconfiguration the ISATAP host calculates an 64 bit interface ID for itself. The fact that this address can not be verified to be unique on the network via multicast, means that the address has to use some sort of unique feature for the host. ISATAP makes the assumption that the underlying IPv4 network is well configured and the client can obtain an unique address. This IPv4 address is then used to make an unique interface ID. The host has now got a link-local address with the prefix `FE80::` that it can use to contact an IPv6 router from the PRL to get the prefix it should use for global unicast addresses. The process can be described in the following three steps.

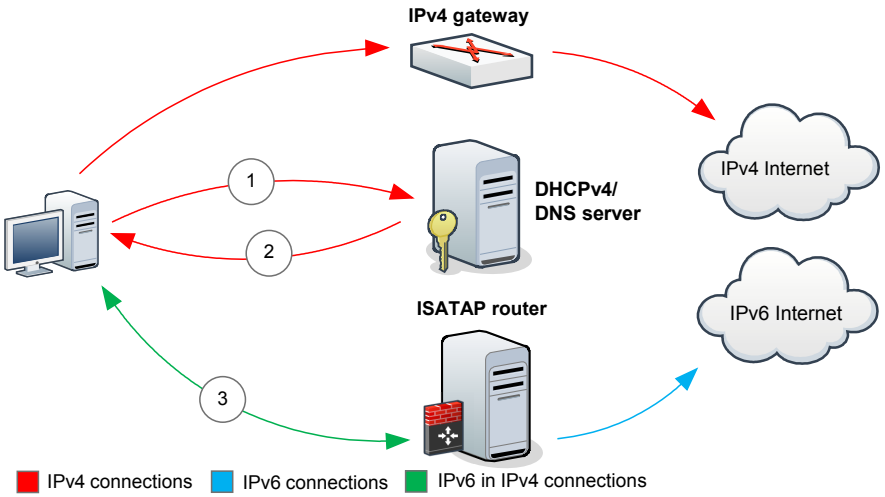


Figure 7: ISATAP configuration

1. The host configures the IPv4 network with the help of DHCP or static configuration.
2. The host will use DNS to build a PRL.
3. ISATAP client uses the 32-bit IPv4 address (`192.0.2.1`), converts it into hexadecimal (`C000:0201`), adds 32 more bits (`0000:5EFE`) and gets the 64-bit interface ID (`0000:5EFE:C000:0201`). A link-local address is formed with the `FE80::/10` prefix (`FE80::5EFE:C000:0201`), encapsulated in an IPv4 packet and used to send a router solicitation message to a host in the PRL. The reply will give the host the prefix it should use for its global unicast address e.g. `2001:db8::5EFE:C000:0201`.

All ISATAP hosts must be dual stacked to function, but the network does not need to be. This means that we are able to get IPv6 access in networks that have networking equipment that does not understand IPv6 packets. As long as the ISATAP client is able to obtain the address of a working ISATAP router and get a useful prefix it can use IPv6 through the IPv4 only networking equipment. When this happens it logically becomes a member of a shared network on IPv6 that

contains all the ISATAP clients and can talk directly to them, and if the ISATAP router has IPv6 Internet capabilities it can use this a the default gateway to the IPv6 Internet. Because ISATAP is adding their IPv4 address in the last 32 bits of the interface ID it does not care what type of IPv4 address it is. In contrast to 6to4 it can use private addresses, as long as the underlying IPv4 network are properly managed the addresses will be unique, and when adding a prefix the clients will have global unicast addresses.

Another big difference to some of the other transition techniques is the fact that ISATAP does not help the user in getting IPv6 connectivity to the outside world. Its only objective is to spread IPv6 connectivity to the clients asking for it, due to the fact that it only do unicasted RA's, but does not care where these clients are in the IPv4 world. This means that it can be used over the Internet, over local network or any other network configuration as long as it can support proto 41 tunnels.

Security considerations

ISATAP can be seen as a more managed version of the 6to4 mechanism. Where 6to4 uses the anycast address and let the network do the rest of the routing, ISATAP needs configuration from the network administrators to function properly. If the clients are unable to obtain a PRL it is rendered useless. This means that spoofing these PRL's would be highly desirable from an attackers point of view and these should be kept updated and secured at all times.

As we discussed in Section 2.3 there are privacy concerns rising from the security community when using the same address over a long period of time. ISATAP with its mandatory way of assigning addresses makes the usage of the privacy extensions to autoconfiguration impossible.

When deploying ISATAP, one should take into special consideration the logical network topology that this adds. All of the clients with active ISATAP connections will effectively form a subnet in your network where, if not configured properly, everybody can talk to each other regardless of the IPv4 setup. This threat has to be addressed when setting up the ISATAP router, making sure that the default routes being set on the clients and the router will keep the existing IPv4 segregation intact.

4.2.5 Teredo

In many of the transition mechanisms we have looked at to this point has some special requirements that will make it difficult to use in existing networks. 6over4 requires IPv4 multicast, 6to4 needs global addresses and ISATAP can not do NAT traversal. Often protocol 41 is also used to encapsulate IPv6 in IPv4. All of these requirements will be a problem on many networks as many IPv4 network devices does not support the mechanisms natively, does not have global addresses, can not do multicast, are using NAT or does not handle protocol 41. This renderers many transition mechanisms useless on networks with one or more of these constraints.

In RFC4380 [54] the author aims to describe an alternative for networks with one or many of the aforementioned obstacles. It aims to mitigate these obstacles by tunneling IPv6 packets inside IPv4 UDP datagrams, since UDP is essential for many important Internet features, e.g. DNS, it is supported in every NAT device and router. He chose to call his protocol Teredo after *Teredo navalis*, a shipworm that bore tunnels through wood. This because, in his own words;

> the animal only survives in relatively clean and unpolluted water; its recent comeback in sev-

eral Northern American harbors is a testimony to their newly retrieved cleanliness. Similarly, by piercing holes through NAT, the service would contribute to a newly retrieved transparency of the Internet.

These features makes Teredo able to work in networks where the devices are totally IPv6 unaware. To do this it needs three specific components to work:

Teredo client A host with IPv4 connectivity to the Internet, running the Teredo protocol to get access to the IPv6 Internet.

Teredo server A well known host that has a global IPv4 address and is reachable for the client. This server is used for the initialization of the protocol and forwards very little traffic for the Teredo client. Its primary usage is to find the closest Teredo relay for the client and to find the clients for the relays.

Teredo relay The tunneling endpoint of the Teredo tunnel. This node is found by the Teredo server and the client uses it as the decapsulation/encapsulation node for the tunneling in the protocol. This fact means that it has to be a dual stack host with global addresses on both IPv4 and IPv6.

The generation of an IPv6 address when using Teredo is reflecting the need for traversing NAT devices in the network. Table 5 is showing an example of how this address will look. This particular address reveals to the server that the client is using the prefix reserved in RFC5156 [81] (`2001:0000::/32`) for Teredo and are using the Teredo server `65.54.227.12`, are behind a cone NAT, using port `40000` for NAT traversal and are behind the public IP `192.0.2.45`.

Table 5: Example on a Teredo IPv6 address

Bits	0-31	32-63	64-79	80-95	96-127
Address	2001:0000	4136:e378	8000	63bf	3fff:fdd2
Decoded		65.54.227.120	cone NAT	40000	192.0.2.45
Description	Teredo prefix	Teredo server address	NAT type	UDP port	Client address

Together these three parts of the Teredo protocol and the generation of the IPv6 address are able to get a tunnel up and running behind almost every statefull firewall and NAT device. For the client to be able to establish this connection it needs a known Teredo server that can handle the initial setup, this setup is a rather elaborate initialisation process which is visualised in Figure 8.

1. The client sends a IPv6 ping encapsulated into a IPv4 UDP datagram to the preconfigured Teredo server
2. The server decapsulates the packet and forwards it to the IPv6 destination
3. Destination host answer with a ping reply to the nearest Teredo relay
4. The Teredo relay that got the reply forwards it to the server it originated from

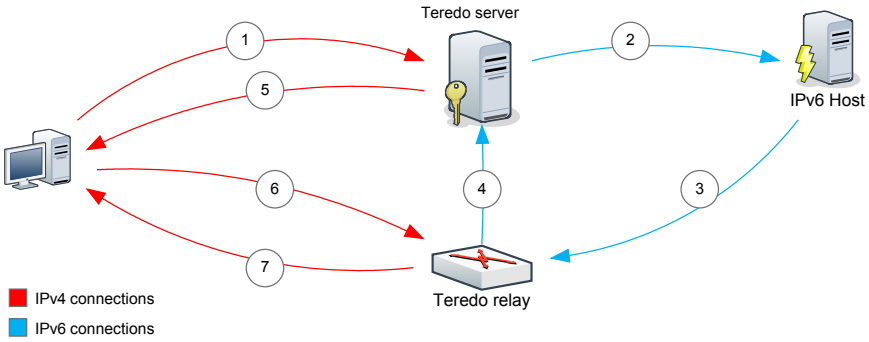


Figure 8: Teredo tunnel setup

5. The server sends back a packet to the client telling it what relay to use

6/7. The client now switches to exchange encapsulated traffic directly with the relay, initiating the connection from inside the NAT network and are thus able to reach the relay and make it possible for it to answer.

As we see from the figure, Teredo is completely dependant on the fact that the client and server is able to communicate. When the Teredo link is idle in between connections the client needs to refresh the opening in the NAT firewall. The mapping will only be kept for a certain amount of time, if this threshold is crossed, the server will not be able to contact the client if an incoming IPv6 connection is arriving. To mitigate this problem the Teredo client is at a given time interval sending what the RFC refer to as *bubble packets*, this is empty packets that has got the sole purpose of keeping the connection alive.

Security considerations

As we have seen in Section 3.1.2 the introduction of NAT on the Internet has broken the end-to-end connectivity that once where a primary goal. Unfortunately we have grown so accustomed to NAT that many people see it as a requirement for securing such networks. When clients behind NAT start to use Teredo, we effectively make them available in the global address space. This means that it can be contacted from every IPv6 host on the Internet. If this is good or bad are in the eye of the beholder, but one thing is certain, network administrators are in trouble if this is not taken into consideration. Newer Windows versions are supporting Teredo and in special circumstances one could find themselves in the position that machines will suddenly punch straight through all firewalls and be world reachable.

The very nature of Teredo is to get out of any network configuration, this gives it a very annoying feature for network administrators. It has no easy way to be stopped. The author of the RFC discusses this and say that one can easily block outgoing IPv4 UDP traffic with destination port 3544 and incoming IPv4 UDP traffic with the same source port. This will stop Teredo from functioning in compliance with the RFC. However, attackers can easily set up their own Teredo server that simply uses another port.

Another problem is the traffic coming directly from the relays, Figure 9 shows how the packet looks when intercepted on our IPv4 interface, just an UDP packet from someone on the IPv4

Internet that is let through the firewall with high indistinct port values. If we however look at the same packet, but let Wireshark [82] treat the UDP data as IP traffic we see the decapsulated IPv6 traffic that has entered our network. The packet in Figure 10 is the same packet and as we can see this is just a harmless ping request. But how would we go about stopping this traffic? The only way would be applying the Teredo protocol on all incoming and outgoing UDP packets to see if they can be recognised and alerts triggered or just simply discard the packets.

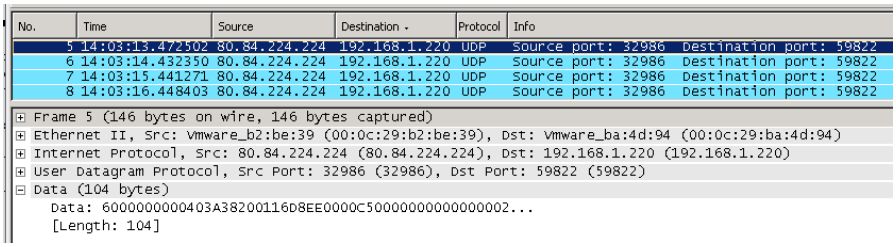


Figure 9: Original IPv4 UDP traffic

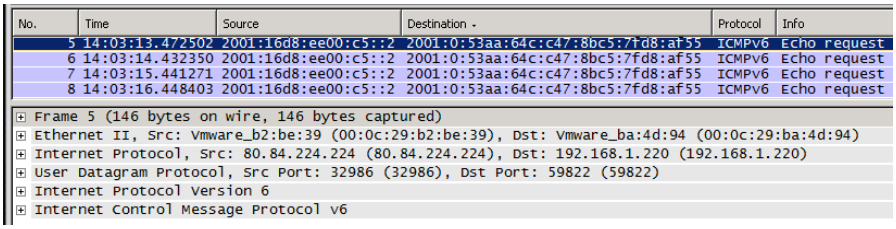


Figure 10: Original IPv4 UDP traffic analysed as IPv6

When comparing to ISATAP there are similarities but also differences. Just like ISATAP, Teredo's strict regime for generating the IPv6 address that is to be used prevents it from applying the privacy extensions and the address itself contains much useful information. Unlike ISATAP it has no way of directly connecting to other Teredo clients in the same internal network, this means that all traffic has to enter the Internet and be routed by the Teredo services in the cloud.

4.3 Protocol translation

In the previous sections we have looked at techniques that aims to get a working IPv6 connection to the network, or single hosts themselves. Protocol translation techniques have a slightly different goal. They assume that an IPv6 stack is available on the host or that the host knows of an host in the network which is dual stacked. There exists two main types of protocol translators, the difference between the two are the placement of the translator in the network. We have got the translators that are situated on the edge of the network and getting traffic from the network and host based types that translates traffic before leaving the host.

4.3.1 Network translators

The network translators are dual stacked known hosts that can translate between IPv4 and IPv6. These are situated on the network in places that enables them to take general IPv4/6 traffic

that has got an IPv6/4 destination and translate between the two with various means. There exists two distinct techniques to do this that has existed for some time, and despite the fact that they are not in themselves a translation mechanism they are frequently reused in new transition techniques.

RFC2765 [83] defines a technique called *Stateless IP/ICMP translation algorithm* (SIIT) to translate directly on the IP-level. It describes a way that the host can take IPv6 traffic, strip away the header and add an IPv4 header and vice versa. This is done by translating Time To Live (TTL) to Hop Limit, Type of Service (ToS) bits to traffic class, payload length is recalculated and possible fragmentation fields can be copied to a fragmentation header. TCP and UDP are virtually unchanged from IPv4 to IPv6 and can be copied directly over. ICMP has got bigger differences, but these issues are solved within the SIIT specification. This means that it can be used to relay traffic between IPv4 and IPv6 only hosts.

DNS Application Layer Gateway(DNS-ALG) described in RFC2694 [84] is another technique that is often reused in newer proposals. DNS-ALG focuses on DNS queries that only asks for an A record. This means that the host making the query is IPv6 ignorant, and can not use it. When such records are found, DNS-ALG will replace it with a query that asks for both A and AAAA records. If the external host is IPv6, DNS-ALG will create a fake A record that are sent back to requester and create a mapping between the IPv4 and IPv6 address.

Network Address Translation - Protocol Translation (NAT-PT) as described in RFC2766 [56] is using ordinary NAT functionality as described in the original RFC [85, 86] together with SIIT to propose a method that would allow IPv6 hosts to share one or more IPv4 addresses. The technique itself is very controversial as it basically introduces NAT into IPv6, by allowing many IPv6 hosts to share one IPv4 address. Many people consider this a step back with regards to the problems IPv6 was set out to solve. These issues are discussed in detail in RFC4966 [60] that argues to degrade it to historic status, and the IETF recommends that it no longer should be used.

At the time of this writing, the IETF has got no drafts that could solve the problems arising when degrading NAT-PT. They do however recognise the need for such functionality and there exists drafts in the process of replacing NAT-PT with the ”Framework for IPv4/IPv6 Translation”[87] and reviewing and improve SIIT with the ”IP/ICMP Translation Algorithm”[88]. Both are still just drafts but are showing that IETF are working to create a solution.

Transport relay transporter (TRT)

The *transport relay transport* [89] is a protocol translator that is thought to be used in IPv6 only networks. This is a dual stacked host that sits at the edge of the network and is meant to service IPv6 only hosts that for some reason have to communicate with IPv4 only hosts.

The main idea behind TRT is to give the IPv6 hosts an address to contact that simulates their existence in the IPv6 address space. This is often by DNS, using the aforementioned DNS-ALG mechanism. When a host is doing a DNS query and the DNS server is only giving back an A record, the DNS-ALG mechanism will create a AAAA record that embeds the IPv4 address in the lower 32 bits of a known prefix that are routed to the TRT host.

When the TRT host is receiving traffic to these special addresses, it knows that they need to be translated into IPv4. The translation itself is on the TCP/UDP level and will keep track of the

state of these connections. The connections will be terminated at the IPv6 side of the TRT host and a new TCP/UDP connection will be initiated to the IPv4 host, and thus relaying the traffic between the two.

To the end nodes, this translation is completely invisible. For the internal IPv6 host everything seems normal because of the AAAA record that the DNS request came back with, and it will assume that it is talking to the external host itself. The external host does not care in the sense that it is just another connection that it serves on the IPv4 network.

Security considerations

The fact that TRT sits on the edge of the network and translates between the two protocols it will disrupt the normal functionality of the packet flow. This is invisible to both the sender and receiver, but are adding some constraints on what they can do. It is also an effectively single point of failure and if saturated with traffic or targeted with a DoS attack it can break functionality in the network.

End-to-end encryption is one of the things that has to be sacrificed with this technique. IPsec is unable to traverse TRT because of the translation on the IP level, it disrupts the flow in such a manner that it can not establish the connection.

DNSSEC is also affected by TRT by the fact that the mechanism uses its very own DNS lookup procedure and inserting fake AAAA records when the desired host only exists on the IPv4 Internet. These records are not signed by anyone and DNSSEC can not verify them[2].

Authentication based on the IP address will also be affected by TRT. Some older remote login protocols will because of this not work over TRT (e.g. *rsh, rlogin*), this is unfortunate as older legacy systems that use these protocols are also the ones that most likely will be unreachable on IPv6. In these cases the options are to get the hosts to run IPv6 themselves or implement newer remote login protocols as SSH that will traverse TRT with no problem.

4.3.2 Host based translators

Both of the techniques described here have the same goal, they aim to make it possible for legacy applications that are IPv4 only to continue to work, even when facing IPv6 only hosts. This is done by intercepting the communication from the application directly on the host before it hits the network and thus not affecting the performance on the network itself at all. This means that they should scale well and the hosts using these techniques will be seen by the network as ordinary dual stack hosts, previously described in Section 4.1.

BIS

Bump in the stack (BIS) [90] aims to provide old legacy software that are not IPv6 aware a way to communicate with IPv6 only hosts. To make these legacy applications able to speak to IPv6 hosts we need to ”trick” them into believing that they are actually talking to native IPv4 host. As we can see in Figure 11 this is achieved by inserting some new functionality into the IPv4 stack on the host.

The mechanism works on the assumption that every application uses a DNS request to acquire the address of the hosts it will contact, this gives the mechanism an opportunity to determine

[2]Which in fact is a feature of DNSSEC, because the records are fake.

if the application in question is IPv6 aware. The *Extension name resolver* looks at the outgoing DNS requests and acts on the ones that only requests A records, modifies these to also include AAAA records and passes it on. If the request comes back with only an A record it will let the application talk directly to the IPv4 stack and get out of the way. Does the reply contain an AAAA record, the connection is passed on to the *address mapper* that give back an IPv4 address that is inserted into an A record reply and given back to the application. Making the application living with the notion that it takes directly to an IPv4 host. This method is based on the previous work done with DNS-ALG, described in Section 4.3.1.

The address mapper keeps an address pool of IPv4 addresses that can be used for connections. When the extension name resolver gets an AAAA record that it needs to talk IPv4 with, the address mapper takes one address from the pool and pairs it with the AAAA records IPv6 address, stores the pair in a table and gives the extension name resolver back an IPv4 address to relay back to the application.

When the previous two modules have done their job, the application has an IP address from the local address pool which it thinks is an external host. It then sends traffic to this address. At this point the translator comes in and translates between IPv4 and IPv6, making IPv4 packets into IPv6 packets and vice versa. To be able to do this, the IP conversion mechanism from SIIT described in Section 4.3.1 is used.

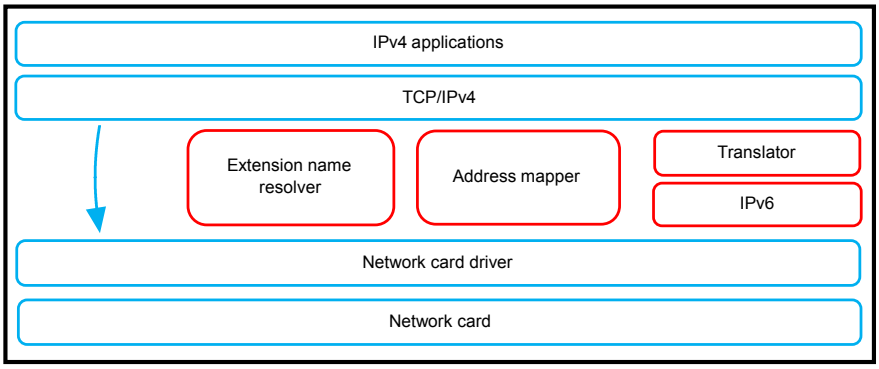


Figure 11: BIS diagram

BIA

A very similar approach to the same problem is proposed in RFC3338 ”Dual Stack Hosts Using "Bump-in-the-API" (BIA)” [91]. They rely on the same basic principle of intercepting IPv4 traffic and translate it to IPv6 when this is a necessity. The difference is that BIA does this on the API level before the program call reaches the IP stack. As we can see in Figure 12, BIA is inserting the modules between the Socket API and the IP stacks and are able to snoop the socket calls and detect the ones it should act upon.

The *address mapper* and *name resolver* have the same functionality as the modules described for Bump-in-the-stack. The difference is the *function mapper*. Like BIS, BIA will only act upon request from programs that have been intercepted by the name resolver and are by their DNS

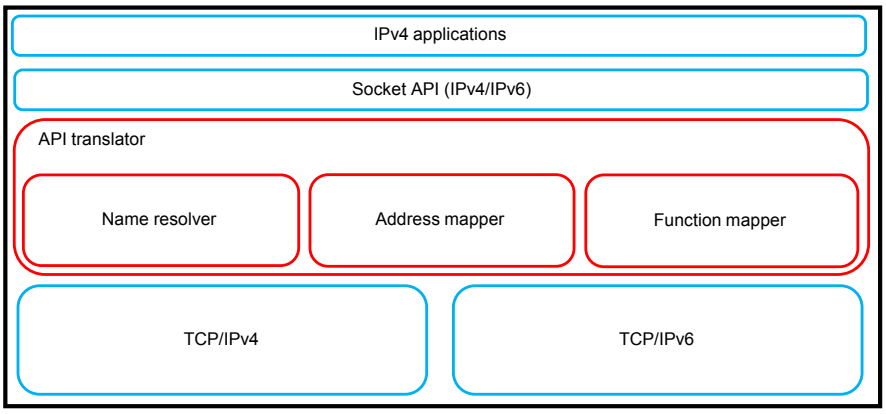


Figure 12: BIA diagram

requests labeled IPv4 only. When this happens, the *name resolver* and the *address mapper* have made a fake A record to be handed back to the application and created a mapping for the IPv4/IPv6 pair that are to be used. Now the *function mapper* can monitor the IPv4 address that has been assigned and intercept socket functions going to that address. When a function call is intercepted, it will get the corresponding IPv6 address, translate the socket call to IPv6 and create the socket against the external host. When traffic is received from the IPv6 host, the opposite happens. The function mapper will intercept the call, ask the address mapper for the corresponding IPv4 address, translate the socket call to IPv4.

Security considerations

Although the two mentioned host based translators have much in common with the network based ones, the fact that they exist on the host itself give them some advantages over the network based ones. First and foremost they keep one of the early dogmas of networking. They keep the network dumb and are doing the work on the end hosts. For all the network is concerned, they are normal IPv6 hosts and are not treated any different than other hosts. This keeps the complexity on the network low and give administrators an easier task of managing the network.

In contrast to some of the network based translators, they have the advantage that they can manage and run end to end security measures on the host. Many of the same mechanisms are in use, but because of the fact that all this is done in software before leaving onto the network it is still end to end.

In bigger networks, the network based translators can become saturated with traffic or targeted with DoS attacks that makes them unusable for the clients and creates a single point of failure. When moving this feature from the edge of the network and onto the host itself you remove this attack vector. The number of hosts running BIS/BIA are not of concern for the network as its does not create any extra load, and by this fact the host based translators should scale very well.

4.4 Summary

As we have seen in the previous sections there exists a multitude of techniques that are designed to help us when doing the transition to IPv6. Table 6 tries to give a quick summary of the different types and to give some insight into the usage of these techniques.

Table 6: Summary of transition mechanisms

	Summary	**Usage**
Dual-stack	Running complete and separated IPv4 and IPv6 stacks on every host	If you have enough IPv4 addresses and the equipment supports it, dual stack everything
Tunneling mechanisms		
Configured tunneling	Using static IPv6 in IPv4 tunnels to get IPv6 connectivity	Use when ISP can not deliver native IPv6 access or you need to bypass some IPv4 only equipment
Automated tunneling	Automatic encapsulation of IPv6 packets with ”compatible addresses”	Obsolete and should not be used
6over4	Using the existing IPv4 network as a link layer for multicasting IMCPv6 messages	Not widely supported and used because of the multicast requirement
6to4	Automatic generation of a /48 network to every public IPv4 address	Used in networks with one assigned IPv4 address to create a globally routable IPv6 prefix
ISATAP	Using the existing IPv4 network as a link layer and using a known ISATAP router for multicast etc	Connecting a set of hosts over IPv6 in an existing IPv4 network
Teredo	NAT traversing IPv6 tunneling through the usage of UDP packets	Used for single hosts behind NAT firewalls that require IPv6
Translation mechanisms		
NAT-PT	Edge mechanism that enables IPv6 hosts to share one or many IPv4 addresses	Not longer endorsed by IETF
TRT	Translating between IPv6 and IPv4 only hosts while keeping UDP/TCP state	Used for IPv6 hosts that need to communicate with IPv4 only hosts
BIS	Translates IPv4 headers into IPv6 headers on the host and vice versa	Making IPv4 only software function over IPv6
BIA	Translates IPv4 socket calls into IPv6 socket calls on the host and vice versa	Making IPv4 only software function over IPv6

But is this the right mindset when it comes to IPv6? The sheer time scale of this ”transition” should make the people in charge of IPv6 in the organisations change their mindset from ”transition” to ”implementation”. Forcing them to consider all the different aspects of adding IPv6 and to see the two protocols as two equals that have to live together for some time ahead. It is only after a long period of running both protocols that we can begin to speak of the transition, and the transition is not IPv6 coming in, but rather IPv4 going out.

No matter how we choose to look at it, we need to start acknowledging the fact that we need to implement IPv6 in a much faster pace then we are today. To help us with this we have the aforementioned techniques that all have different weaknesses and strengths. Dual stacking has the advantage that we logically separate the two protocols and run two different stacks on every host, getting an easy-to-understand networking setup in return as it can be viewed separately.

The disadvantage is that with dual stacking everything on the network, we do not use any less IPv4 addresses and must in many cases keep NAT devices. Dual stacking is also dependant on our upstream ISP are able to offer IPv6 access to the site.

The tunneling protocols have one thing in common, they all aim to communicate with IPv6 over the underlying IPv4 network. The scope of the different ones range from Teredo that are best suited for single hosts and all the way to 6to4 that provide every IPv4 host with a official IPv4 address a /48 network with 2^{80} globally routable IPv6 addresses. This means that in many cases these protocols will be used in cooperation with dual stacking, the tunneling protocols providing the IPv6 access, and dual stacking organises the internal network.

The protocol translation techniques are still in what can be called their infancy, these protocols are thought to be used in IPv6 dominant networks that do not run IPv4 themselves. This will with the current dominance of IPv4 be techniques that are very heavily used and can run into performance issues if deployed too early. When organisations are phasing out IPv4 support they will come back, giving IPv6 only networks the possibility to communicate with IPv4 only sites. As a technique used to implement IPv6 now, they are just not suitable.

4.5 Selected techniques

To do the rest of this thesis with all the aforementioned protocols would be impossible with the time frame at hand. To be able to perform systematic and thorough tests we have chosen three protocols that we will look at further. Of the techniques that are discussed we have chosen three tunneling techniques. This is due to the fact that the tunneling techniques are the best suited for adding IPv6 to IPv4 networks, many of the other techniques have the focus of getting IPv4 to work in predominately IPv6 networks. Such networks networks are very seldom and we think the need to add IPv6 at this time should be the priority.

When choosing techniques to continue to work with in the thesis we have chosen a set of three techniques that all have different scopes of operations. 6to4 is a protocol that aims to give the user a /48 prefix for every global unicast IPv4 address and does because of this it can cover both the WAN and LAN side of a network. ISATAP is more focused on giving the user a flexible way of distributing IPv6 on the internal network. The last technique chosen is Teredo that does kind of the opposite of ISATAP, its only focus is getting single hosts access to the IPv6 Internet. These three will give us similar but different approaches to the problem at hand and hopefully help us enlighten different aspects of this problem.

Native IPv6 has been present for a very long time in every operating systems that you look at, but the addition of different transition mechanisms have only become available in the last years. Because of the large market share that Microsoft has over the desktop market[3] and the fact that Microsoft has in the recent years become a driving force for the deployment of IPv6 we have chosen to use the three techniques that are supported out of the box in Windows 7 for further work in this thesis. These techniques are 6to4, Teredo and ISATAP. Fortunately all the techniques are also supported in the Linux distributions we have chosen to use and because of this we will not have problems with unsupported clients.

[3] According to Wikipedia 91.6% at the time of this writing.

5 Planning the implementation

In this chapter we will describe the equipment we have used for our baseline network and explain the configuration of the different parts of the baseline network. When we have set the premises for the baseline network we will plan the different transition mechanisms that were chosen in Section 4.5. This has been the chosen way of doing things as it lets us make educated decisions on the deployment before anything is actually done to the running network.

5.1 Building a baseline network

To conduct the experiments on how the different transition techniques will work in existing IPv4 networks we needed an existing IPv4 network that would reflect a somewhat typical setup for a business network. In this section we will describe the hardware used in the testing and how the software were configured to create the baseline network.

5.1.1 Physical hardware

In the experiments that we are conducting, we are basing the whole setup on virtualisation technology from VMware [92] as the basis for our lab. VMware is a reputable vendor of virtualisation platforms and the fact that the network is virtualised should not affect the functionality. Our main VMware ESXi 4 installation is being run on an HP Proliant DL585, the specifications can be seen in Table 7. This will enable us to make a testing base with the exact same IPv4 features and settings on the hosts involved for every test scenario. It will also ease the workload on resetting the base network. Snapshots are taken on each machine after the installation of the Virtual Machines (VM) and all the updates from the operating system vendor has been applied. When one of the individual tests has been performed and the results have been extracted, we can then roll back the complete test network and easily get back to the same starting point for the next scenario.

Table 7: Physical hardware

Name	**Configuration**
Software	VMware ESXi 4.0 server
Processors	4 x AMD Opteron 850 1.8 GHz
Memory	8 GB Registered ECC
Storage	4 x 36,6 GB in RAID 5
Network	NC7782 Dual 10/100/1000 Mbit

5.1.2 Network

All networking on the VMware machine is handled by the internal virtual switches of VMware. The server itself has got two physical network interface cards (NIC), but for our testing we will only utilise one that are connected to the infrastructure here at Gjøvik University College. NIC 0 is connected to a port configured with IPv4 only access and is getting a unfiltered globally

routable address via DHCP and are thus considered as an IPv4 only ISP. The firewall has been configured to terminate the link from the ISP on its WAN-port and use NAT to share it internally via the LAN interface. It uses DHCP to distribute IP-addresses in the internal network and is running a local DNS-server that caches local records and forwards the rest to the DNS-servers acquired from the DHCP client running on the WAN-interface. The technical details are shown in Table 8 and the corresponding configuration file can be viewed in Appendix A.1.1.

Table 8: m0n0wall setup

	Description
Firewall	
Hostname	m0n0wall
IP	192.168.1.1
DHCP	
Address pool	192.168.1.100 → 192.168.1.199
Lease time	7200 seconds
Options	Default gateway and DNS server at 192.168.1.1
DNS	
Local domain	example.com
Local hosts	Manually entered A records for the local servers; *debian-server.example.com*, *win-server.example.com* and *m0n0wall.example.com*
Forwarding	Local domain queries are kept locally, all other requests are sent to the DNS servers acquired by DHCP on the WAN interface
DHCP hosts	Local hosts that states a hostname in their DHCP query are added to the DNS server
Routing	
Incoming rules	Default deny
	Packets with source addresses from RFC1918 networks are dropped
	Packets with destination port 22/80/3389 are allowed
	Packets with established state are allowed
Outgoing rules	All outgoing packets with source address from the internal network is allowed

To simulate an ISP that delivers both IPv4 and IPv6 access we have a specially configured firewall that does configured tunneling as described in Section 4.2.2. By using a tunnel broker[78] we get access to a /48 IPv6 prefix. This tunnel is set up with the built in support for the tunnel brokers client in the m0n0wall project, the tunnel is a ”Anything in anything (AYIYA)” [93] tunnel that gives us the tunnel endpoint `2001:16d8:ee00:c5::2` and the IPv6 prefix `2001:16d8:ee71::/48` that are routed to the endpoint. This enables us to simulate the existence of IPv6 from our ISP on a border router that is not handling IPv6 traffic. The IPv4 configuration is the same as for the IPv4 only set up and the corresponding configuration file from m0n0wall can be seen in Appendix A.1.2.

The first machine that handles the ”Internet connections” is a m0n0wall firewall. Two such firewalls are set up and are fed the external link on their WAN port as the connection to the Internet. The internal LAN port of these firewalls are then connected, one at a time, to a logical VMware vswitch, giving us the opportunity to easily switch the clients from an IPv4 only ISP

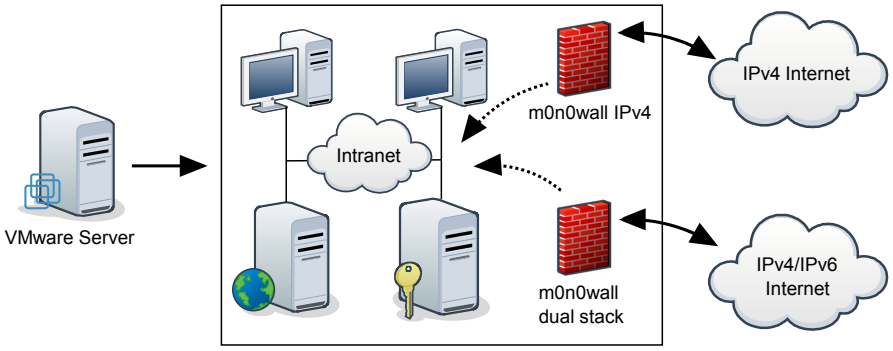


Figure 13: Baseline network hardware setup

to a dual IP setting. This is required as the different transitional mechanisms have different requirements and are meant to fulfil different purposes, some intend to get IPv6 connection from the cloud while others are distributing an already existing connection. This setup is visualised in Figure 13.

5.1.3 Software

The baseline network is setup with two different client operating systems and two different server operating systems. Specifications for the different VM's can be seen in Table 9.

When designing our baseline network we have chosen to use two different workstation operating systems and two different server operating systems. This is done to replicate a small modern network with different varieties in the machines connected. We have chosen to use recent versions of all the operating systems for our experiment. This choice is based on the fact that the newest stable releases today are the ones that are being assessed for implementation when bigger changes of equipment are planned and their functionality should be tested before implementation.

The networking is kept simple and aims to represent the fact that most small/medium sizes businesses uses a NAT setup, typically a dedicated router that runs all their networking services and does the necessary firewalling. To control the internal networks we have used m0n0wall [94] as the firewall, this firewall is free software and is based on the Unix clone FreeBSD. m0n0wall is a very lightweight firewall distribution that aims to be a router/firewall and nothing more. This gives us an uncomplicated firewall that we can rely on to do only what we ask from it and not have other ”features” disrupt the operation.

Services on the internal network are hosted by two different VM's, this way we can get both Windows and Linux on the server side. The Linux distribution used is the newest stable release from the Debian project [95], 5.04 at the time of download. On this Linux server we are currently hosting a web site with the help of the Apache [96] web server and a SSH login possibility with the help of OpenSSH [97]. Both server daemons are the latest stable version downloaded from the Debian projects repositories. The Windows server of choice is the Windows Server 2008 [98]. This server is used for a comparison of the IPv6 handling capabilities of the two platforms, to enable us with some remote testing we have enabled the remote desktop feature of the system

Table 9: Virtual machines

	Version	CPU(s)	Memory	Network
Firewalls				
m0n0wall IPv4	1.31	1	128MB	
WAN				DHCP
LAN				192.168.1.1
m0n0wall IPv4/IPv6	1.31	1	128MB	
WAN IPv4				DHCP
WAN IPv6				2001:16d8:ee00:c5::2
LAN IPv4				192.168.1.1
LAN IPv6				2001:16d8:ee71::1
Servers				
Debian	5.04	1	128MB	192.168.1.210
Windows Server	2008	2	2GB	192.168.1.220
Clients				
Windows	7	2	1GB	DHCP
Ubuntu	10.04	1	256MB	DHCP

and opened the respective port in the firewall.

On the client side we have also decided to implement operating systems from Microsoft and a Linux distribution, ideally we should have an Apple computer but because of the impossibility to virtualise that operating system this had to be excluded. The Linux distribution used is the Ubuntu [99] distribution, this free operating systems is the most used Linux desktop system. Microsoft's desktop operating systems should not need an introduction as they are the market leader and has been since the mid 90's. Windows 7 [100] is the most recent version and is deployed in our base network. Both operating systems have been installed with the default configurations and have got all the available updates applied. In addition the Windows 7 workstations have got Microsoft's own antivirus, Security Essentials, installed to mitigate possible virus infections. No configurations beyond these are applied to the clients for the baseline network. Visualisation and configuration can be seen in Figure 14.

5.2 Implementation planning

With the baseline network in place we will in this section focus on research question 3 from Section 1.5. The biggest problem when setting out solve complex problems is to loose the clear overview of the situation. As described in Section 1.6 we have chosen to use a divide and conquer tactic to mitigate this threat. We have divided the planning phase into 5 different parts:

Goals for network operation In the first part of the planning we look at the technology at hand, evaluate the theoretical functionality as described in Chapter 4 and set some goals for how we want the network to operate when fully implemented. This will help us when planning the alterations and the evaluation when the implementation is finished.

Alterations - Firewall When we have decided how the network should work we go into planning the alterations that we have to employ to get the desired functionality. We start with the

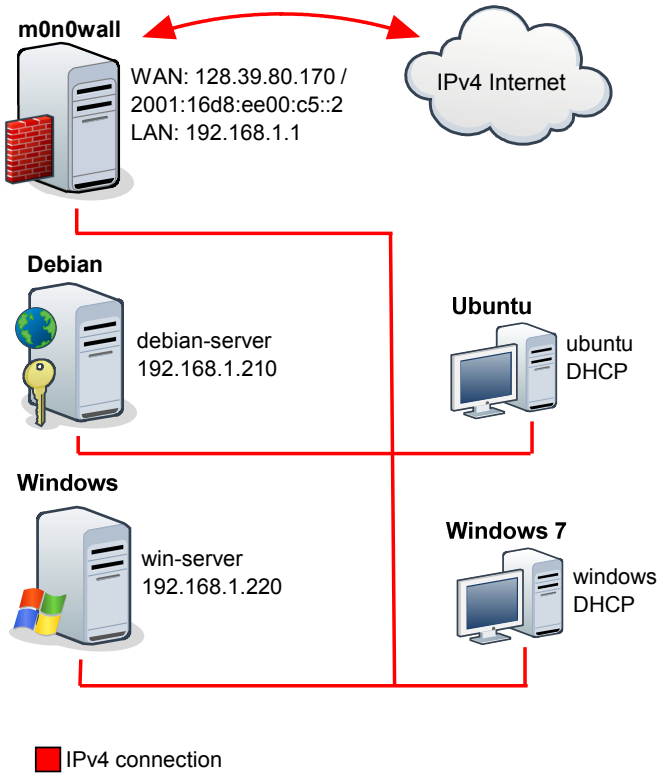


Figure 14: Baseline network IP setup

device that are handling our connection to the Internet and internal network services. To make sure that the steps described are working, we insert tests that have to be passed before the next part is started. With doing the firewall first we are sure that the internal and external requirements for connection is fulfilled before any network hosts are altered.

Alterations - Servers With the connectivity in place we continue by altering the hosts that are servicing clients both internally and externally. The same testing is planned here to make sure that the part is functioning according to plan before continuing.

Alterations - Clients When the two previous parts have gotten the connectivity and the services running on IPv6 it is time to let the ordinary clients in. This part of the planning will describe this and add testing to make sure that they work as planned.

Final testing Although the previous parts includes testing along the way, we decided to do a last verification of functionality before the planning/implementation can say to be finished.

If these steps are followed, we should be able to plan the different parts without loosing the total overview. The testing in the different parts will make sure we have certain points in the implementation that will need to be fulfilled before we continue. If some of the tests fail we can then halt the implementation and evaluate if we should continue or abort, giving us what we can call multiple points return.

In the following sections we will make plans for each of the transitional technologies and give the reader a clear understanding on how the implementation is conducted.

5.2.1 Implementing 6to4

The first transition mechanism we will have a look at is 6to4. We will try to emphasise the requirements it demands to operate and what kind of limitations that puts on our deployment. After deciding what will be deployed we will set up a detailed plan for the implementation.

Requirements

As described in Section 4.2.3 6to4 has some requirements that has to be met when using this transition technology. They can be summed up as a few key points:

- The 6to4 client must use a globally routable IPv4 unicast address
- The network must support the usage of protocol 41 tunnels
- The ISP must handle traffic to the anycast address `192.88.99.1`

Like many businesses in the real world our baseline network has got one official IPv4 address and uses NAT to share this between the connected machines. This means that our internal clients can not run 6to4 as one requirement is an official IP address. We then have one plausible scenario left and that is to deploy 6to4 on the edge of the network, since 6to4 equips every official IPv4 address with a /48 prefix we should have enough addresses to use for the internal client machines.

To do this we need to set up the *6to4 router* on the edge of our network and use it to terminate the 6to4 traffic from the outside world and deliver the internal network with native IPv6 connectivity by sending out router advertisements to enable the clients to utilise autoconfiguration to set up their networking interfaces with IPv6 connectivity.

To help us to this we are in the fortunate position that m0n0wall supports 6to4 on the WAN interface, and can also configure the internal interface with an IPv6 address from the /48 prefix acquired by 6to4. This means that the adding of 6to4 does not involve adding extra hardware in the network but we can use existing devices that already are tested and functions properly.

Goals for network operation

The servers in our network will be configured with a static IPv6 address and added to the DNS server with the corresponding AAAA records. A server should be static in its nature and this will also help the administrators to manage the network and separate the different types of hosts. Clients will also benefit from this as the address for the serving machines always will be available on the same address and make it possible to create static routes in the firewall to allow access to services from outside connections.

To allow a network without address conflicts and manual configuration of every host we will enable the firewall to send out RA messages as described in Section 2.3, this enable the clients to configure their own addresses with the assigned prefix, get a default route and make sure that it does not come in conflict with other hosts on the network.

When the clients have got an IPv6 address autoconfigured we basically have got an IPv6 network up and running. All the clients and servers will have an address and a default gateway configured. To let the hosts that support it receive additional options about the network we will also enable DHCPv6 on the firewall. In cooperation with the RA messages with the O-flag set it will provide the clients with the DNS-servers IPv6 address to make the internal requests use IPv6. As a result from the clients having DNS over IPv6, we have a goal that the better part of the internal network traffic will be going over IPv6 as all the servers will have a AAAA record.

The default routing on the network will have the same base characteristics as the IPv4 counterpart. The biggest exception will be the removal of NAT. Default deny incoming traffic except for the wanted services will be applied to the WAN side and only allow established connections entrance.

Planned alterations

For the implementation of 6to4 to be successful in our baseline network we will need to work out a detailed plan before the actual changes are made. These plans are made on such a level that they are clearly separated and are done individually. This will enable us to abort the implementation in an early stage if something is not functioning properly thus creating points where tests are run and have to be successful before continuing.

In this part we will try to sum up the most important changes that will be made to the different hosts in the network. The order of the following subsections creates a list to work out of when doing the experiment and the testing in each subsection will be carried out before moving on to the next step.

Firewall

The firewall in our base network has to be configured with a 6to4 client on the WAN side of the device. This will establish a connection to the nearest 6to4 relay via the anycast address `192.88.99.1`. When this is confirmed and working, we enable IPv6 on the LAN side of the firewall. This is possible by using one of the subsequent addresses from the prefix assigned. Another test to verify the connectivity from the LAN interface will be done before releasing IPv6 onto the network.

After successfully testing the connectivity via both the WAN and LAN interfaces we will move on to enable RA messages onto the network. This will enable the rest of the hosts to autoconfigure their IPv6 addresses and get the necessary route onto the IPv6 Internet. Confirmation of the activation of autoconfiguration will be collected before we turn on the last part of the network configuration, namely DHCPv6 and adds the O-flag to the RA messages to tell the clients that extra configuration is available.

Servers

Our two servers will be the next natural step to take in adding IPv6. We will add the support for IPv6 to the running kernel, set static IP addresses on their interfaces, add the default gateway, add IPv6 DNS server and finally testing that the IPv6 connectivity is functioning properly with regards to both the servers general connectivity but also that the services it is running can support and receive traffic over IPv6. The daemons handling the services offered may in many cases need to be configured to allow for traffic over IPv6 and this will be checked.

Clients

The different clients are the last to be enabled with IPv6 and we will now enable IPv6 autoconfiguration and DHCPv6 on their network interfaces and let them configure themselves. When we have verified that they are configured as they are supposed and can communicate over IPv6 we can move to last point in the implementation.

Verify services

The final step in the implementation is to do a verification process on all the different services running in the network to make sure that everything that is working on IPv4 and can support IPv6 also works on IPv6. The AAAA records available from the DNS server should make the clients use IPv6 when contacting the local services. This will be tested both on the internal network, but also the external services offered will have an AAAA record and be tested for functionality.

When all services have been tested both internally and externally, all clients have got IPv6 enabled TCP/IP stacks and all the servers are up and running we can call it a success and consider the implementation finished.

5.2.2 Implementing ISATAP

The second transition mechanism we will have a look at is the ”Intra-site automatic tunneling addressing protocol”. We will list the special requirements this transitional technology have for successful operation before planning the detailed steps necessary to make it function properly in our network.

Requirements

As we have seen in the previous section 6to4 enables the network to gain access to address space from the IPv6 Internet thus eliminating the necessity to receive a native IPv6 connection from your ISP or in other ways get IPv6 connectivity to your border. Like we described in Section 4.2.4, ISATAP has a somewhat more limited scope in its functionality. The aim for ISATAP is delivering a way to distribute an existing IPv6 connecting to either a subset of hosts in the network or hosts that are located at places in the network that does not support IPv6.

This means that ISATAP demands that the ISATAP router has got access to a IPv6 global unicast address but leaves it up to the implementer to determine the best way of arranging this. This can be done with an ISP that supports native IPv6, configured tunneling or even 6to4, the important thing is that the ISATAP router gets an IPv6 prefix it can deliver as response to incoming RS messages.

The potential router list is the key to make ISATAP function properly. Clients are counting on that a DNS query to a given host will give them such a list. A well known DNS hostname will be needed for the distribution of this list and will be made a static A record in the DNS server.

Goals for network operation

When our network is deployed with ISATAP we will have one logical ISATAP subnet. These are in reality IPv4 only parts of the network that are assigned an IPv6 prefix, but in our experiment the whole network will utilise ISATAP. To handle the ISATAP routing we will have to implement a routing device in the network which will be the default route for hosts trying to reach IPv6 Internet. Traffic going to other ISATAP hosts will in many cases be sent directly between the

ISATAP hosts themselves.

To segregate the ISATAP hosts from the rest of the network, we will put them on their own subnet. This will give us the oppurtunity to make the rules in the firewall more specific and enabling us to establish a static route that routes all traffic with an ISATAP host as their destination directly to the ISATAP router.

Like 6to4 the clients will use autoconfiguration, but instead of the regular IPv6 multicast messages they will use unicasted RS messages sent directly to the ISATAP router to get the prefix and default route.

The DNS service could be run on IPv6 in this case, the m0n0wall firewall that handles the service is available over IPv6, but the fact that we have to add another single point of failure (the ISATAP router) to get it to work, makes us choose to use IPv4 for this service.

Planned alterations

For the implementation of ISATAP to be successful in our baseline network we will need to work out a detailed plan before the actual changes are made. These plans are made on such a level that they are clearly separated and are done individually. This will enable us to abort the implementation in an early stage if something is not functioning properly thus creating points where tests are run and have to be successful before continuing.

In this part we will try to sum up the most important changes that will be made to the different hosts in the network. The order of the following subsections creates a list to work out of when doing the experiment and the testing in each subsection will be carried out before moving on to the next step.

Firewall/ISATAP router

The firewall in this scenario is handling the routing to and from the IPv6 Internet for the ISATAP router. The clients must contact the ISATAP router to enter the IPv6 Internet. This means that the firewall must forward IPv6 traffic with the appropriate destination address to the ISATAP router when it enters on the perimeter of our network, and open the necessary ports for machines hosting external services.

The fact that our firewall also is running the DNS service requires that we enter the A record for our ISATAP router, namely `isatap.example.com`. Without this entry the clients will not know where to retrieve their *possible router lists*.

Our firewall in this case do not support ISATAP routing out of the box and we need to introduce a new service to the network. In this case we have chosen to use the new Ubuntu 10.04 server for this purpose, this edition is what the Ubuntu publisher Canonical calls a Long Term Support (LTS) release and will because of this ensure that proper support is available if needed. This host will get a static IPv4 and IPv6 configuration. This means that it can reach both the IPv4 and IPv6 Internet. With the addition of an ISATAP interface it can communicate with IPV6 over the existing IPV4 infrastructure and send out RA messages to the other clients in this setup making them able to autoconfigure their ISATAP interfaces.

Servers

Windows server does in this configuration support ISATAP by default and since we are using the the host `isatap.''DNS suffix''` which are the default setting in Windows server 2008 we should

not have to configure the interface. The connectivity will be verified and that the hosted services are available over ISATAP IPv6.

The Linux kernel has since version 2.6.25 supported ISATAP and can be easily configured to use this feature. We will need to add a virtual interface and configure it to use the correct ISATAP router for it to function properly. Fortunately there exists a package called `isatapd` that handles the creation of these virtual interfaces and should make our job easier. With this installed we need to verify the connectivity and test the hosted services so we confirm that they work over IPv6.

The fact that ISATAP in its nature uses the existing IPv4 addresses when generating their interface ID makes the configuration of static IPv6 on the servers redundant. As long as they have static IPv4 addresses they will also generate the same interface ID every time and thus getting the same IPv6 address.

Clients

In recent versions of the Ubuntu operating system, a program package called `isatapd` has been available through the official repositories. This daemon, according to the maker, ”creates and maintains an ISATAP client tunnel” in compliance to RFC5214. This will be used on our Ubuntu clients to handle the client configuration.

Windows 7 is supporting ISATAP out of the box and will by default try to configure the ISATAP interface from `isatap.''received DNS suffix''`. This means that it should be able to obtain an IPv6 address as soon as the ISATAP router and A record are added to the network without any added configuration.

Verify services

The final step in the implementation is to do a verification process on all the different services running in the network to make sure that everything that is working on IPv4 and can support IPv6 also works on IPv6. The AAAA records available from the DNS server will make the clients use IPv6 in contacting the local services. This will be tested both on the internal network, but also the external services offered will have an AAAA record and be tested for functionality.

When all services have been tested both internally and externally, all clients have got an IPv6 address via ISATAP and the testing has been successful we can call the deployment a success.

5.2.3 Implementing Teredo

In this section we will plan the implementation of the Teredo tunneling mechanism. We will list the special requirements this transitional technology have for successful operation before planning the detailed steps necessary to make it function properly in our network.

Requirements

When comparing Teredo to the other two techniques we have planned it differs quite a lot from them. As we saw in Section 4.2.5, the main focus of Teredo is to get the host an IPv6 connection from behind almost every NAT device. The requirements for Teredo to function can be summed up in the following short list.

- Every host on the network must have a Teredo client capable of performing the necessary

steps according to the RFC

- The network has to be connected to the Internet and every host must have a default route to access it
- The network must allow outgoing UDP traffic to port 3544 which are used for the initialisation of the protocol

This means that we do not need to do much to our network configuration. The network will as is support the creation of Teredo tunneling and will not affect the performance at all.

Goals for network operation

In the nature of Teredo every host in out network will be a Teredo client. This means that we do not differ between the servers and clients in the planning. The topology of Teredo is very simple and every host with an active Teredo interface will just be an end node on the IPv6 Internet.

The fact that the address can change every time the client reinitialises the connection also makes it unsuitable to host services on. If we wanted to host things on a Teredo based connection it would require a DNS updater that adds the correct address when the client changes IPv6 address. Such services exists on the Internet today [101] but we have decided not to use them as this is both impractical and slow as these clients check for changes after a certain threshold. Based on these facts we have decided that the network should function as normal on IPv4 and host services there and primarily use the IPv6 connectivity to connect to external IPv6 only services. We will however test the different services to check the connectivity from external sites and to verify if the services can be run over Teredo.

Clients

The Windows based computers in the network are supporting Teredo natively and do not need any configuration to start using their interface. Microsoft has some limitations on how these interfaces can be used, and we need to look into these limitations.

None of the Linux based distributions in our network supports Teredo without installation of a client. Fortunately this exists in the repositories of both Ubuntu and Debian and by installing the `miredo` package we should be able to get a virtual interface like the one used in the ISATAP experiment. This interface are treated like any other interface and will be used with no hesitation to IPv6 enabled hosts.

Verify connectivity

As we are not going to serve any services over IPv6 we will finish of the implementation of Teredo by verifying that all the clients can use IPv6 only hosts in the Internet by checking the DNS resolving of AAAA records, pinging the hosts and actually using the connectivity to test the service.

6 Experiments

In this chapter we will perform the planned actions with regards to the transition mechanisms described in Chapter 5. We will go through the steps needed to perform the implementation, the practical problems that we encountered and sum up the effects of the alterations so they are clear before the discussion in Chapter 7. The different mechanisms will be performed and finished one at a time in Sections 6.1 / 6.2 / 6.3 respectively.

6.1 6to4 transition

The deployment of IPv6 in the baseline network has been done according to the planning done in Section 5.2.1. This means the steps described there have been followed and the alterations and testing have been performed. In this section we will describe the implementation and the difficulties we faced and what was done to overcome them.

6.1.1 Firewall (pre-servers)

The first step needed is to enable IPv6 support on the firewall in general. When this is done we moved on to configuring the WAN interface. Fortunately for use m0n0wall has support for the 6to4 protocol out of the box and this just a matter of saying that the WAN port is to use 6to4. We confirmed that the interface had gotten the IPv6 address of `2002:8027:50aa::1` with a full /48 prefix available.

When the WAN interface was confirmed successful with the acquiring of an IPv6 prefix we started to configure the LAN interface. We decided to use the automatic functionality to take the first address from the the first subnet in the prefix acquired by the WAN interface, this gave us the address `2002:8027:50aa:1::1/64`. This was confirmed by the status page for the interfaces showing the correct IPv6 address for the LAN interface.

m0n0wall does not come with IPv6 enabled by default so the routing table on the firewall needs the adding of some rules to make the connection useful. As noted in Table 8 on page 42 the default setting for connections is deny, this is also true for IPv6. We added a rule to allow traffic from the LAN interface access to the Internet and the established connections back to the sender. With this modification done we were able to verify the connectivity by pinging the IPv6 only server `ipv6.google.com` from the firewall itself.

Problems

In this phase we ran into some problems when we confirmed the 6to4 address on the WAN port, but were unable to use it in any way to get connectivity to the IPv6 Internet with the built in ping tool in the web interface. After researching the Internet we found that other people on the m0n0wall forums had the same problem and the explanation was that for some reason the ping attempts would not leave the firewall unless the LAN interface also had IPv6 enabled. After turning on IPv6 on the LAN interface the connectivity tested OK.

6.1.2 Servers

As according to the planning, our servers should be manually configured with static addresses. We started with the Debian server and edited the `/etc/network/interfaces` file. This file is run every time you start the networking subsystem on the host and we added the address (`2002:8027:50aa:1::10`), netmask (`64`) and default gateway (`2002:8027:50aa:1::1`). To make sure the the system uses IPv6 for its DNS lookups we also added the address of the m0n0wall host in the `/etc/resolv.conf` file. Both files can be seen in Appendix A.2.

To make sure that the configuration would stay static we rebooted the server before testing connectivity and availability on the hosted services. We were able to ping IPv6 only hosts on the Internet, connect to the SSH daemon running on the host, connect to the local web server and verify that DNS queries were using the IPv6 address of the DNS server on the m0n0wall host. This was done with the tools; `ping6`, `ssh`, `lynx` and `dig` respectively.

The windows server was configured in a similar matter. Manually setting the static settings for the connection with the address (`2002:8027:50aa:1::20`), prefix length (`64`), default gateway (`2002:8027:50aa:1::1`) and the preferred DNS server set to IPv6. After a reboot we did the same test on connectivity and availability on the remote desktop hosted on this machine. Both machines were 100% problem free.

6.1.3 Firewall (post-servers)

After the servers were configured and we had verified that the services they hosted were functioning over IPv6 we had to add some more functionality on the firewall. The first thing was to make sure that the DNS server in the firewall would resolve the local servers with both an A and AAAA record. This was done by manually adding the AAAA record to the local DNS cache and test that it worked with a simple DNS query for any type of record from every host and verify that we were served both types.

After this was checked OK, we moved on to make sure that the services that should be world reachable were let through the firewall. This meant opening port 22/89/3389 for incoming traffic with destination `2002:8027:50aa:1::10` for the first two ports and destination `2002:8027:50aa:1::20` for port 3389.

These two alterations were the last things needed to make the servers fully functional over IPv6. The next step is now to enable the autoconfiguration so the clients can get a proper IPv6 address with this technique. We enabled the router advertisements on the LAN interface and set the O-flag so clients with a DHCPv6 client knew that more configuration was available. The DHCPv6 server was also enabled, given an address pool and set to give out the IPv6 address of the DNS server.

Problems

The adding of AAAA records to the local DNS server gave us some trouble. When we tried to add the AAAA record for a hostname that already existed with an A record, we were denied by the web interface. After consulting with the m0n0wall community we found that this was a known issue and the bug was fixed in the 1.32 version that were released the 17. of march 2010. We upgraded the firmware on the firewall and verified that both records were available at the same time.

When setting up the LAN interface we started by using the setting that allows the LAN interface to be automatically configured by the prefix used on the WAN interface. This would be very helpful if the ISP is running DHCP on the WAN interface or just decided to give us a new IPv4 address. Unfortunately this is not possible due to the fact that the DHCPv6 server requires a static IPv6 address before it allows you to create an address pool to give out addresses from. It turned out that this bug can be circumvented by first setting the LAN address static and configure the DHCPv6 server. When the configuration has been tested, switch back to automatic 6to4 on the LAN interface and it will configure itself if a IPv4 address change will occur.

6.1.4 Clients

The last portion of the network is to configure the clients in the network. We started with the Ubuntu client that were already configured via autoconfiguration and had gotten a default route to the IPv6 Internet. We checked the settings in the network manager and added the DHCPv6 option for IPv6. With these settings it should be automatically configured to enable the preferred useage of the network.

On the Windows 7 clients we did not have to configure anything. They autoconfigured themselves and got the correct DNS server from DHCPv6. Tested the connectivity and DNS resolving over IPv6.

Problems

After configuring the Ubuntu client we noticed that it did not get the IPv6 address for the DNS server via DHCPv6. After some searching on the Internet we found that this is a problem with the current version of *Network manager* (v0.80) that is shipped with the version of the GNOME project that Ubuntu uses. This is a shortcoming in the version as it currently does not support DHCPv6. The needed support is on the road map for the next release and will by then hopefully support DHCPv6. All other tests on the client ran successful but all DNS request went over IPv4. Although this were a failed test for this client we decided to continue with the implementation as the client could resolve DNS over IPv4.

6.1.5 Testing and Summary

At this point all the steps from the planning are carried out and the tests along they way have been successful. We now verify everything with a set of tests that will check if the connectivity and availability is at the level we expect after the adding of IPv6. The tests that have been run can be seen in Table 10, and the visualisation of the new network structure in Figure 6.

After doing this implementation we feel that 6to4 is a mechanism that is very close to implementing native support for IPv6. The internal network is running in complete dual stacked mode and do not know of the 6to4 mechanism working on the edge. This creates as we have seen a considerably amount of configuration on the hosts in the network and the network components that are being used. In this special case when the edge has to do the 6to4 it also requires the network to be IPv6 aware and IPv4 only hardware will have to be changed. On the other hand, the switch to native IPv6 will be easier as the implementation in the network has already been done and all you basically need to do is switch the prefix being used.

Table 10: 6to4, tests performed

DNS resolving	Result
All machines confirmed able to resolve AAAA record for `ipv6.google.com`	**OK**
All machines confirmed able to ping IPv6 only host `ipv6.google.com`	**OK**
All machines confirmed able to resolve the local AAAA records	**OK**
Internal services	
SSH available from all internal clients on `debian-server.example.com`	**OK**
Web available from all internal clients on `debian-server.example.com`	**OK**
Remote desktop available from all internal clients on `win-server.example.com`	**OK**
External services	
SSH available from external client on `2002:8027:50aa:1::10`	**OK**
Web available from external client on `2002:8027:50aa:1::10`	**OK**
Remote desktop available from external client on `2002:8027:50aa:1::20`	**OK**

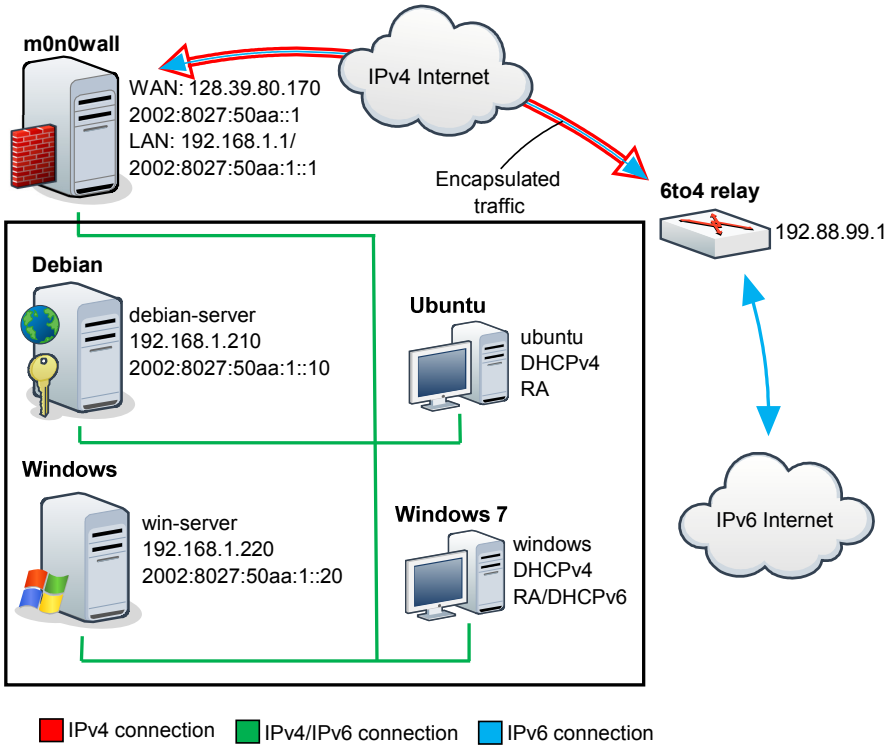


Figure 15: The network with 6to4 implemented

6.2 ISATAP transition

The deployment of IPv6 with ISATAP in the baseline network has been done according to the planning in Section 5.2.2. The steps described there have been followed and the alterations and testing have been performed. In this section we will describe the implementation and the difficulties we faced and what were done to overcome them.

6.2.1 Firewall

In Section 6.1.3 we discovered that the release of m0n0wall used in the baseline network had a bug with handling A and AAAA records for the same host. This means that to ensure the proper handling of these records we upgraded to version 1.32 that was confirmed to work in the previous experiment.

The firewall in this scenario had terminated the IPv6 connectivity from the ISP on it's WAN port. We confirmed that an address had been set to the interface, configured the internal LAN interface with the static IP `2001:16d8:ee71:1::1` and added a rule in the firewall to allow traffic with source address in the `2001:16d8:ee71:/48` prefix access to the IPv6 Internet.

To make the ISATAP clients able to find the unicast address of the ISATAP router we added the A-record for `isatap.example.com`. We also decided that the subnet `2001:16d8:ee71:2::/64` should be assigned for the ISATAP hosts. This meant that we had to add a static route on the firewall so that all traffic with the destination of the ISATAP subnet was routed to the ISATAP router.

As we described in Section 4.2.4 the client generation of an interface ID is dependant on the IPv4 address. This means that we can calculate the IPv6 addresses the servers will get and open the corresponding ports in the firewall to allow the external services over IPv6. Port 22 and 80 where opened for traffic with the destination of `2001:16d8:ee71:2:0:5efe:192.168.1.210` and port 3389 was opened for traffic with destination `2001:16d8:ee71:2:0:5efe:192.168.220`.

6.2.2 ISATAP router

The first thing we did was to install the addition to the baseline network. We installed Ubuntu 10.04 server with all the standard choices, updated after installation and were ready to configure the necessary settings.

Our first priority was to get the machine working against IPv4 and IPv6 Internet. To make it possible to establish static routes to the host this needed to be statically configured. The host was given the IPv4 address of `192.168.1.230` and IPv6 of 2001:16d8:ee71:1::10. This was done by altering the file `etc/network/interfaces` which can be viewed in Appendix A.2.2. We verified the connectivity to the Internet for both IP versions and moved on to make it work as a gateway for ISATAP clients.

We used the `isatapd` package with the default settings to create the virtual interface `is0`. This is a interface that uses the built in support of ISATAP in the Linux kernel and sets up a interface that tries to autoconfigure the interface. This is not what we want so we have to configure this statically as well. Due to the fact that `is0` is created after the `interface` file is used, we use the file `/etc/rc.local` to set the options we need on this interface. The IP should follow the ISATAP standard and are in the previous chosen subnet, i.e. `2001:16d8:ee71:2:0:5efe:192.168.1.230`. In the same file we also activate IPv6 forwarding to be able to route packets onto the Internet and restart the autoconfiguration daemon. The file can be seen in Appendix A.2.2

Now we have a host that is able to reach the IPv6 Internet, can talk ISATAP and can route packets. All that is missing is the service that responds to the client router solicitation messages. This is handled by `radvd`, as the name implies a Router Advertisement Daemon. This is set to run only on the `is0` interface, deliver the prefix `2001:16d8:ee71:2/64` and only do unicast requests.

This means that it will not actively advertise the route but only answer to unicast requests from the clients. The configuration file for the daemon can be seen in Appendix A.2.2

6.2.3 ISATAP clients

We started with the Windows server as this version supports ISATAP natively, the server was rebooted to ensure that the the ISATAP client we reloaded and we verified that it got the desired prefix and could reach IPv6 only sites on the Internet. We tested remote desktop over IPv6 and verified that it was connectable over IPv6.

The Debian server also supports ISATAP via built in features of the Linux kernel. In the same way as on the ISATAP router we used the `isatapd` package to create the virtual ISATAP interface and confirmed that the interface was able to autoconfigure itself. The services hosted on this machine were tested and we found that SSH was working normally but the Apache web-server could not serve sites over the virtual interface. The error was identified as a problem occurring because the Apache web server was started before the ISATAP interface was ready configured and was not able to bind to this interface. After adding a restart of the Apache web-server in the `/etc/rc.local` file enabled it to recognise the `is0` interface and serve web pages over IPv6.

The rest of the ISATAP clients (Windows 7 and Ubuntu 10.04 desktop machines) were pretty much configured the same way as the servers. The Windows 7 was just rebooted and verified working on the IPv6 Internet, the Ubuntu machine got `isatapd` and verification of successful installation was concluded when it reached the IPv6 Internet.

6.2.4 Testing and Summary

When we have reached this point in the experiments we have been able to implement all the necessary changes to the network. We now have to verify that the changes work from different points of view. Both for internal hosts, but also the external services that are meant to be world reachable. The tests we have run can be seen in Table 11 and the visualisation of the network can be seen in Figure 16.

Table 11: ISATAP, tests performed

DNS resolving	Result
All machines confirmed able to resolve AAAA record for `ipv6.google.com`	**OK**
All machines confirmed able to ping IPv6 only host `ipv6.google.com`	**OK**
All machines confirmed able to resolve the local AAAA records	**OK**
Internal services	
SSH available from all internal clients on `debian-server.example.com`	**OK**
Web available from all internal clients on `debian-server.example.com`	**OK**
Remote desktop available from all internal clients on `win-server.example.com`	**OK**
External services	
SSH available from external client on `2001:16d8:ee71:2:0:5efe:192.168.1.210`	**OK**
Web available from external client on `2001:16d8:ee71:2:0:5efe:192.168.1.210`	**OK**
RDP available from external client on `2001:16d8:ee71:2:0:5efe:192.168.1.220`	**OK**

To summarise the implementation of ISATAP is a mixed experience. On one hand, the implementation worked flawlessly and provided the clients with an IPv6 connection that have the nice

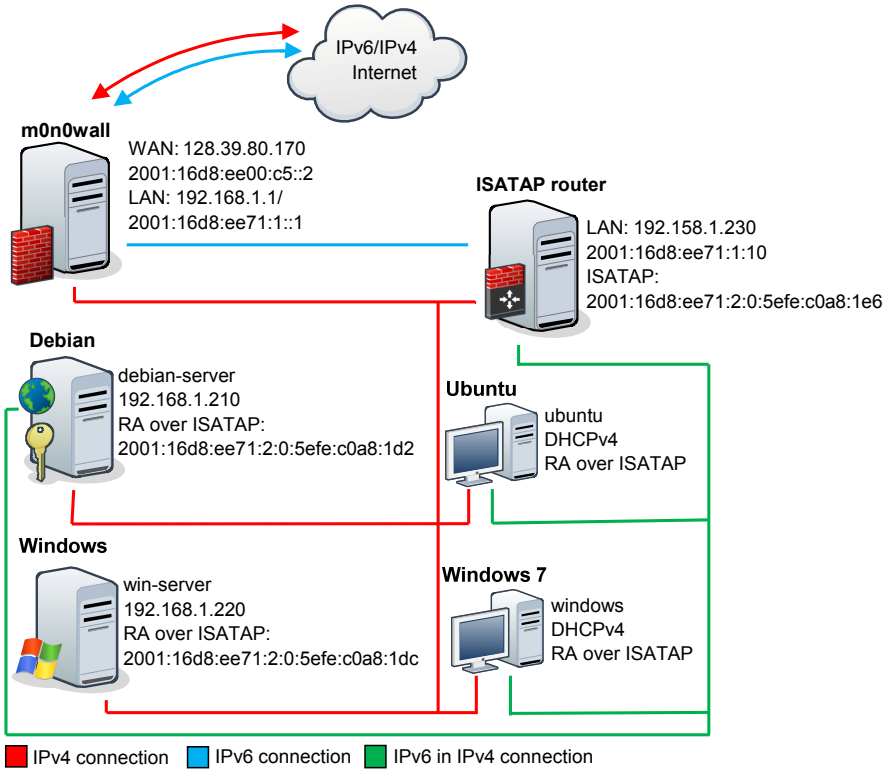


Figure 16: The network with ISATAP implemented

feature of working over IPv4 only equipment. On the other hand, ISATAP does not give you any IPv6 connection to the Internet and would because of this fact require that such a connection is established. The cost of then implementing IPv6 native in the network with the existing IPv6 connectivity versus the cost of getting ISATAP to function would be an interesting scenario to look at. There are many variables in such a calculation, but it should be done before implementing ISATAP.

6.3 Teredo transition

The deployment of IPv6 with Teredo in the baseline network has been done according to the planning done in Section 5.2.3. The steps described there have been followed and the alterations and testing have been performed. In this section we will describe the implementation and the difficulties we faced and what were done to overcome them.

6.3.1 Teredo clients

We started with the Debian server. The `miredo` package was available through the official Debian repositories and after installation we verifired that a new virtual interface had emerged, namely the `teredo` interface. This had configured itself an address that started with the teredo prefix of `2001:0` and seemed to be configured properly. The process was repeated on the Ubuntu host and both machines were verified with a proper Teredo IPv6 address, and the ability to reach IPv6

only sites on the Internet. All the services running on the Debian host was also tested, both from the Ubuntu client and a external IPv6 host, and found to be working very fine over the Teredo interface.

The Windows machines in this case should be the easiest ones to set up for usage with Teredo. Both Windows Server 2008 and Windows 7 supports the feature out of the box, but will not use the connection like any other ”normal” IPv6 connection. Due to the nature of Teredo and how it pokes holes in the firewall, its usage is heavily restricted. The Windows DNS-client will not ask for AAAA records when the host has got only link local or a Teredo based address. This makes it impossible to use Teredo to surf the web and makes it a very seldom used connection.

There are ways of tricking the DNS client, by adding a non link-local address to a interface and add the default `::/0` route to exit on the Teredo, one can make Windows use Teredo as a more normal IPv6 connection. This dirty solution was tested but found to problematic to maintain over time. When the machines rebooted, the address was intact on the interface but the added routes was gone. Windows also recognised the added IPv6 IP as a native connection and according to default policy it does not start the Teredo client when a native IPv6 connection is present.

Microsoft has decided that there should be no easy way of letting the users choose to use Teredo as a real IPv6 connection and although there exists a dirty hack to make it work it is not ready to be set in production. We then have the choice of disabling the Teredo client or keep it around for the rare cases when we want to contact literal hosts, meaning we have to type in the whole IPv6 address for ourselves. There are multiple scenarios where this could be practical, so we keep the interfaces activated.

After getting the Teredo interfaces up and running on Windows and discovering all the limitations that where set on them we were not surprised when incoming connections also was put under a strict regime. All unsolicited traffic entering on the Teredo interface are discarded and we need to specially add exceptions into the Windows firewall to make services available over IPv6 via the Teredo interface. When this was done however, the terminal service on the server was world reachable.

6.3.2 Testing and Summary

When we have reached this point in the experiments we have been able to implement all the necessary changes to the network. We now have to verify how the changes work from different points of view. Both for internal hosts, but also the external services that are meant to be world reachable. The tests we have run can be seen in Table 12 and as we can see it consists of fewer tests than the previous ones. This is because of Teredos scope of operation, it does nothing for the internal network and many of the tests run in Section 6.1.5 and Section 6.2.4 would be utterly useless as Teredo does not aim to provide these services. A visualisation of the alterations on network can be seen in Figure 17

The summary of doing the Teredo implementation can be summed up very easily. The Linux machines are using the Teredo interface as a normal network interface and had no problem using the IPv6 connectivity. It works when communicating with the IPv6 Internet and also on requests that comes in to the hosted servicesn making it work like any other connection. Windows on the other hand have got a bigger problem with Teredo, when you use it in the rare cases when

Table 12: Teredo, tests performed

DNS resolving	Result
All machines confirmed able to resolve AAAA record for `ipv6.google.com`	**OK**
All machines confirmed able to ping IPv6 only host `ipv6.google.com`	**OK**
External services	
SSH from external client on the current Teredo address of the Debian server	**OK**
Web from external client on the current Teredo address of the Debian server	**OK**
RDP from external client on the current Teredo address of the Windows server	**OK**

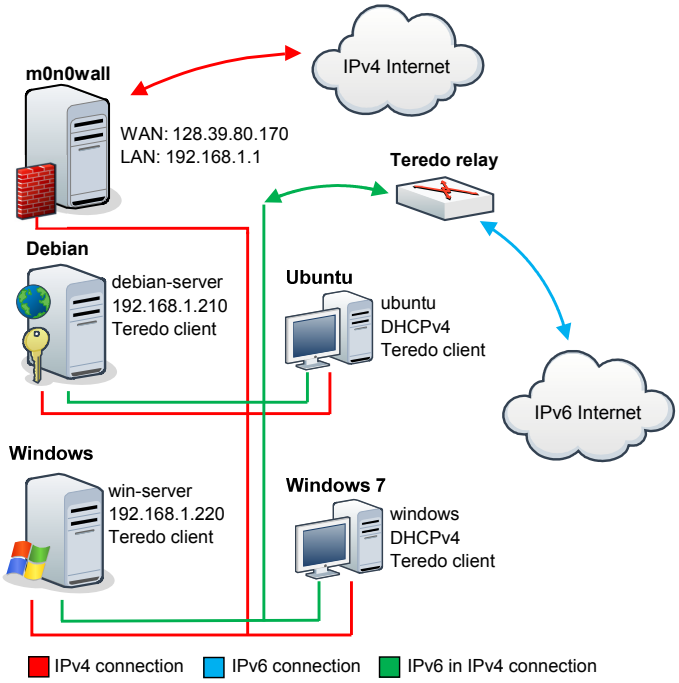


Figure 17: The resulting network with Teredo

it is allowed, it works very well and gives you a stable connection to the IPv6 Internet. But for all other cases you can not use the possibility, this leaves out many people from being able to use IPv6 without configuring extra services in their network. From a security point of view what Microsoft has done is very reasonable but they should add a way of using it in a more normal matter for users that know what they are doing and wants to use Teredo to gain access to the IPv6 Internet.

7 Discussion

In this chapter we will sum up the findings from the combined job of planning the transitions in Chapter 5 and the execution of the planned actions in Chapter 6. We will look at the alterations made in doing the implementations, how these affected the network flow, how the security in our baseline network was influenced and review the findings.

7.1 Doing the analysis

In Section 1.5 we posed research question 3 that set out to make us able to study the experiments and draw some results from implementation experience and the alterations done to the baseline network. To do this we have divided the network in parts to reflect the sub-questions posed in Section 1.5 and the planning done in Chapter 5. This gives us more limited sets of data to discuss and we can easier see the impact the alterations has had on the network.

All of the selected three techniques has undergone the same process of analysis and should because of this be easier to read and clearly see the difference between them. We start with an examination on how the internal network has been affected by the adding of IPv6, in this context the LAN side of the firewall is regarded the internal network. We continue with the external interface (WAN), discuss how LAN and WAN alterations has influenced the firewall operations and end with a summary where we discuss the overall affect the transition mechanism has had on the network.

7.2 6to4

In this section we will analyse the findings from the experiment with adding 6to4 to our baseline network. We will look at how the different places in the network is affected (LAN, WAN and Firewall) and finish with some thoughts around how this transition mechanism affects the network and its operation.

7.2.1 LAN alterations

As previously stated in Section 5.2.1 there exists two ways of deploying 6to4. The edge device that do the routing for the entire network, and the option where every host configures 6to4 by itself. In our case the former is used. For our internal network operation we use the commonly known techniques to configure the devices in our network (autoconfiguration, DHCPv6 and static). This means that the internal network is not really using 6to4 as a transition mechanism, they get an IPv6 address from the central router that happens to be in the range reserved by IANA for 6to4 operation. How this work on the edge and onwards are to them uninteresting.

In Figure 15 on page 56 we can see that the network has not undergone topological changes. There has been no rearranging of physical cables but merely changes in the logical operation of the network. From the previous network running only IPv4 we have implemented a native IPv6 configuration on the LAN. Every client has been dual stacked and talks both IP versions. This fact

gives the internal network some desirable features.

Not dependant on internal IPv4 The complete dual stacked environment gives us an advantage in the sense that the internal network is not dependant on IPv4 to function. If the DHCPv4 server were to crash or malfunction in such a way that the IPv4 connectivity is broken for the hosts in the network, this would not affect the IPv6 operation if the network. All internal services available over IPv6 will still function while their IPv4 counterparts is down. This gives us redundancy in the operation of the internal services, but also adds to the complexity of the network as we have two ”full” IP installations to maintain.

Easier transition to native IPv6 The second big advantage of doing the 6to4 connection on the edge is that very much of the job of implementing IPv6 to the internal network will be done in this phase. The difference between a native IPv6 deployment and this type is the prefix we are getting on the WAN side of the Firewall. When our ISP is deciding to give us a native prefix it is a small job of reconfiguring the statically configured hosts and routes, before changing the prefix advertised by the autoconfiguration daemon. The autoconfigured clients will be presented with a new RA message and just start using the new prefix.

After the switch, all the clients should resolve DNS over IPv6 and prioritise AAAA records over A records when they come across dual stacked hosts. Though this was the intention of the network, we saw in Section 6.1 the Ubuntu client was not able to utilise the DHCPv6 server deployed and automatically get the IPv6 DNS resolver. This means that the Ubuntu machine will use IPv4 for the DNS requests. This has no effect on the DNS operation for the clients themselves as DNS is not IP dependant and will work the same way over both IP versions and give the same A and AAAA records as the clients running over IPv6.

These kinds of irregularities are not something we can completely prepare for and were quite unexpected to encounter, especially when taken into consideration that DHCPv6 is described in the official IETF RFC’s the first time in 2003. The expectation is that this feature is implemented and functioning 7 years after. This does however prevent us from migrating the DNS server to IPv6 only and is a good example for the need to dual stack the network to avoid loss of service. This is something that we should have in the back of our mind when doing the implementation of IPv6. We should not put ourselves in such a position that we can not fall back on the tested working setup. Only when the dual stacking has been active for a considerable amount of time and auditing shows little or no IPv4 activity towards a service should we prepare to take it offline on IPv4.

7.2.2 WAN alterations

When activating 6to4 on the edge node it will become, as described in Section 4.2.3, a 6to4 router. This means that it will take the prefix of `2002::/16`, add the hexadecimal value of its current global unicast IPv4 address and create the /48 prefix that every address can get with this technique. If this is a success we will get a complete /48 prefix which we control and can use in our network. This feature of the 6to4 mechanism should be very attractive for network

administrators as it gives them the opportunity to manage the rest of the network to suit their needs with the vast address space available in a /48.

After getting the prefix we could anticipate that we would need to configure default routes and gateways for IPv6. This is something that has been taken care of in 6to4 by the designers. All traffic going out on IPv6 will be sent to the anycast address of `192.88.99.1`, by the nature of anycast this will be sent to the nearest 6to4 relay that answers to this address and we do not need to configure how the edge router should reach the IPv6 Internet.

A change to the WAN side that needs to be taken into consideration when looking into logs and other auditing tools is the special cases that 6to4 routing has got and the routing that is done when these cases are found. There are two sets of special cases, one for the outgoing packets and one for incoming packets.

Sending traffic to native IPv6 When sending to a global unicast address, the border router should encapsulate the traffic in IPv4 and send it to the 6to4 relay anycast address. This will get the packet to the nearest relay, the relay will decapsulate it and forward it to its final destination on the IPv6 Internet.

Sending traffic to another 6to4 site When the edge router gets traffic that has the destination with the 6to4 prefix there is no need to encapsulate and decapsulate the traffic via the 6to4 relays. In these cases they should encapsulate the traffic in an IPv4 packet according to the standard and send it directly to the IPv4 address derived from the destination address where it should find the corresponding 6to4 router for the prefix.

Receiving traffic from a native IPv6 host Traffic that comes to us with these addresses are sent via a 6to4 relay. We have to decapsulate the IPv6 package and forward it to the firewall for closer inspection.

Receiving traffic from another 6to4 site These packets are telling us that the sender is a site that uses 6to4 itself and according to the protocol they should send the packets directly to the IPv4 address embedded in the IPv6 address. This gives a chance to check the validity of the package. If the IPv4 source address on the encapsulation package is the same as the source IPv4 address we can derive from the IPv6 source address in the encapsulated IPv6 address we forward the IPv6 packet to the firewall.

The effect of the two special cases with the outgoing packets is that we will see packets sent directly to IPv4 addresses that we do not find matches for in the traffic coming from the local LAN. These requests will be IPv6 requests and ”translated” before shipped out on IPv4. Remembering this fact will help administrators when debugging with packet dumps and other tools looking on both sides of a router to check traffic. All other IPv6 traffic being routed will be routed to the nearest relay and thus be sent to `192.88.99.1`.

The encapsulation/decapsulation job that the router performs before sending the packet to the firewall or out in the Internet also gives us some added rules that should be applied in the process. These rules will help us dropping packets that are clearly malformed or pretends to be from senders that are just not possible, this will ease the traffic load on the firewall and create an extra layer of security for our network. The rules can be seen in Table 13

Table 13: 6to4 routing rules

Rules	Explanation
Incoming	
Drop packets with reserved IPv4 addresses in tunnels or 6to4 prefixes	6to4 can not use non unicast addresses
Drop packets from other 6to4 sites where the addresses do not match	IPv4 address in 6to4 prefix should match senders IPv4 address
Drop packets from relay with source address in the 2002::/48 prefix	According to standard, relays should not be used when sending between 6to4 sites
Drop traffic to other than your own 6to4 prefix	It is clearly not for your clients
Outgoing	
Drop packets going out to non global unicast addresses	Non global unicast are not to be sent to the Internet
Only send traffic to 2002::/48 addresses directly to corresponding router	This according to the RFC

If we take these guidelines and rules into consideration when setting up 6to4 on our WAN interface we should be able to make the interface deliver IPv6 traffic to the firewall that is not affected by the fact that 6to4 has been in use. The firewall setup can then focus on the IPv6 aspect of the network.

7.2.3 Firewall alterations

In our case with the implementation of the edge router we should make sure that the decapsulation of the traffic is done before it is inspected by the firewall. This can be done both logically and physically. The 6to4 host can be on a single host with the only purpose of getting the 6to4 prefix and handling this traffic while sending all traffic to the firewall, but it can also just be a virtual network card that decapsulates before the firewall is invoked. Both of the solutions are equally good as long as the firewall is made IPv6 aware.

If we were to terminate the 6to4 connection after the firewall inspection, we would most likely loose control with the IPv6 traffic flow into our network. In our setup the firewall does not know of the 6to4, but only gets IPv6 traffic to inspect. This means that it will have to have two rule sets, one for each IP version. With this is in place we can control the flow coming in both over IPv4 and IPv6 and differentiate the rules based on different needs.

In our case this meant first and foremost to make the firewall aware that something called IPv6 would be coming into the firewall and was traffic that was expected and should be handled accordingly. When the firewall is IPv6 aware we should set up some default rules for this protocol, as previously mentioned we now have two different sets of firewall rules to manage. They are similar in the sense that both sets are based on statefull inspection and should let outgoing traffic from the addresses we have on our internal network pass and their established counterparts back in. They differ however in how this is achieved, on IPv4 the firewall has to use NAT to make it work for multiple clients at the same time. The IPv6 rule set does not have this added complexity and can filter on a host level. Where as the IPv4 part must keep an NAT table over the existing

connections, the IPv6 has got a table over the established connections for every host in the network solely based on their global unicast addresses. This gives us lesser room for error and in bigger networks we can not run into the problem with full NAT tables.

For our hosts that have got services that are meant to be world reachable the routing will also be finer in its granularity. We no longer have to do port forwarding through the NAT gateway but can simply add a rule that says *"traffic on port 80 with the destination address of our web server should be allowed"*. In contrast to the way we do it with NAT, we then have to make DNS pointers that all maps to the same IPv4 address, port forward the specific port through NAT into a host on the internal network and in the last part actually allow this to happen in the firewall rule set. Not only will IPv6 be easier to manage, but also creates easier-to-read firewall configurations and gets a less complex network setup.

7.2.4 General 6to4 considerations

The setup with 6to4 relays and 6to4 routers to make this transition mechanism work has created some new methods to not only attack 6to4 sites, but also native IPv4 and IPv6 sites. These attacks have been released by IETF as an RFC [10] and should be read by all implementers of 6to4. It contains a set of special rules that 6to4 routers and 6to4 relays should follow. If these added security checks on both routers and relays were added in every implementation, most of the attack further described in the RFC would be mitigated.

The fact that 6to4 is basing its functionality on the IPv4 address is telling us that problems will arise if we are getting the connectivity from the ISP by DHCP. When the unicast IPv4 address changes, the IPv6 prefix will also change. If this is not accounted for when implementing 6to4 it will in many cases break connectivity for services, DNS records must be changed and statically configured hosts must be reconfigured. Autoconfigured clients will most likely not notice this change as they just use the new prefix. To prevent this scenario it should be made sure that a static IPv4 address is available from the ISP. This will make sure that the 6to4 prefix stays static as long as the IPv4 address is the same.

The heavy influence the specific IPv4 address have, can also make trouble for networks that have redundant Internet connections. If the above fact is not taken into consideration you could find yourself in a scenario that the primary Internet connection is for some reason down, you switch over to the secondary line but use another IP address towards the world and thus breaking your 6to4 connectivity. When deploying 6to4 in such scenarios we must then make sure that the IP of the network not will change regardless of the line being used and thus keeping the 6to4 connectivity.

Under normal circumstances 6to4 will work fine, but in the cases when network errors are starting to emerge we have a problem. In the nature of the anycast addresses we do not know where our outgoing packets are sent and have no control over the relays that are being used at that point in time. Even worse is the incoming traffic, it can come from virtually any 6to4 relay and if the 6to4 relay, as many of them do, use the source address of `192.88.99.1` it is virtually impossible to work with. The debugging job will because of this be very difficult if not impossible as we do not have one provider we can report the problems to and the route can change without our knowledge. The result of this will be that in many cases we can not address routing issues

ourselves but have to wait until, for some reason, it is fixed or routed via another 6to4 relay.

The close correlation between the functionality of 6to4 and the 6to4 relays are also something that must be considered. At this point in time, the traffic on the relays are managed quite nicely by the infrastructure and we get good latency, short traceroutes and low packet loss. But what happens if suddenly the usage of 6to4 is starting to grow rapidly, will the relay routers be ready for the traffic? As we can see at the site BGPmon.net [102] which monitors AS that advertise the 6to4 prefix. This list is not very long, only 20 advertisements, and if the traffic is growing rapidly we can only wonder if the service providers will manage to keep up with the demand. As an end user this will degrade your IPv6 connection and give you a connection with high latency, low throughput and most likely high packet loss.

7.3 ISATAP

In this section we will sum up the findings from the experiment with adding ISATAP to our baseline network. We will look at how the different places in the network was affected (LAN, WAN and Firewall) and sum it up with some thoughts around how this transition mechanism affects the network and its operation.

7.3.1 LAN alterations

In Figure 16 on page 59 we can clearly see that the internal network has had some radical changes done to the logical way of operating, and exchanging traffic. ISATAP use the IPv4 network as a layer 2 protocol and just like IPv4 use Ethernet frames to transport in this network, ISATAP hosts will use IPv4 packets as their ”Ethernet frames”. To achieve this IPv6 in IPv4 tunneling is used, these packets are identified with the protocol number of 41 in the IPv4 header. This means that we as the figure shows get 2 logical networks on the same physical infrastructure. One that handles IPv4 and one logical network on top of that with all the ISATAP hosts communicating with each other and the ISATAP router. This of course adds to the complexity of the network and will make the job of debugging errors in the network more complicated.

The administration of this second network is handled by, in our case, a new host introduced in the network, namely the ISATAP router. This host is in charge of communicating with the rest of the clients over IPv6, send RA messages and be the gateway to the IPv6 Internet. To do this it was statically configured with IPv6 to be able to communicate with the firewall, and we delegated a prefix it advertised to the clients. According to the ISATAP standard, the ISATAP router should assume that only unicast is available on the underlying network, this means that the RA messages will only be sent by request and should not be seen on the network if not especially asked for by the clients.

Unicasting only of RA messages is giving us the possibility to strictly control which clients that can use ISATAP as no autoconfiguration can be done without the clients explicitly asking for it. For the clients to be able to reach the ISATAP router, we must add the router to the PRL. This is done by DNS and the default setting in both the Linux and Windows version is resolving the host `isatap.your.domain` if not configured otherwise. If you just want all your clients to get ISATAP up and working, you add this DNS record and the default settings will do the job. This worked very well under our experiment. If you however will have a finer control of which hosts that are

getting IPv6 connectivity, a switch to a non-standard name of the router and only the clients with the correct configuration will get the connectivity.

This technique of giving the clients special DNS records to resolve can also be used if we want to segregate the IPv6 hosts into different logical networks. It can be called VLAN'ing of ISATAP, but this will create even more complexity and work to be successful in holding the two networks separated and make sure that the clients belong to the ”correct” VLAN. Segregation into different subnets over ISATAP can be used to differ the different IPv4 VLANs for easier administration, but this will not add to the security of the network in any significant matter. The VLAN segregation done in the IPv4 part of the network will not be compromised over ISATAP. The ISATAP clients regards all members of the same prefix as being on the same network as themselves and will encapsulate and send the traffic directly to the host in question. If this host then is on another VLAN and can not be contacted over IPv4, the ISATAP client will not be able to contact it over IPv6 either.

Contrary to the previously discussed use of 6to4 as a transition mechanism, ISATAP is completely dependant on the underlying internal IPv4 network to function properly. The configuration, as seen in Section 4.2.4, are done over IPv4, the IPv6 addresses are based on IPv4 addresses, the traffic is sent via IPv4 packets and the potential router list is based on IPv4 DNS requests. If IPv4 is damaged or misconfigured, it will take down the ISATAP network as well. The ISATAP router itself will be able to reach the Internet via the statically configured IPv6 link to the Firewall, but loose all contact with the ISATAP clients. This fact makes migrating internal services over to IPv6 only via ISATAP can be a bad decision as all internal traffic going over IPv6 is in reality an IPv6 in IPv4 tunnel and will have added overhead and just add another point of failure for the network.

This is not all negative the fact that ISATAP uses IPv4 info to configure all the IPv6 values means that all design decisions made for IPv4 will be present in the IPv6 part of the network as well. All machines with static IPv4 addresses, typically servers, will because of this fact also be statically configured over IPv6. All IPv6 addressing structure will be predictable and can be added to DNS etc. without any IP configuration of the clients.

7.3.2 WAN alterations

In contrast to the previously discussed 6to4 mechanism, ISATAP does not help us in any way of getting an IPv6 connection to the Internet. It will only help us set up the internal network to work over IPv6. In its name (Intra-Site Automatic Tunnel Addressing Protocol) it even suggests that it could be used to connect different sites over IPv6, we will see that this is a plausible scenario and can in this regard be said to give this ”other” site IPv6 connectivity but it will always have to get a IPv6 gateway it can route the traffic through.

In our network we had an IPv6 connection terminated at the WAN side of the Firewall. This connection was not in active use but was delivered and had gotten an IPv6 address with a /48 prefix going with it. At this point the WAN interface could handle IPv6 but would silently drop all connections. After the implementation we had to handle the IPv6 connection of the ISATAP router itself, by adding a IPv6 address on the LAN interface and enabling the forwarding of traffic to the IPv6 Internet we had opened just enough to let the ISATAP hosts out. That got the packets

from the client out to the IPv6 Internet, but not back in. After adding a static route in the firewall to send all traffic with destination to the ISATAP subnet to the ISATAP router we got a working configuration, and basic IPv6 connectivity for the clients.

7.3.3 Firewall alterations

The firewall was partially IPv6 ignorant in our baseline network. It had an IPv6 connection on the WAN side, but did not use it for anything and had not even configured IPv6 on the internal interface. No ruleset existed for IPv6 and it did not do any routing on IPv6, every packet was dropped. This meant that the firewall was IPv6 aware, but had been intentionally configured to deny IPv6 traffic access to our network. As we saw in the previous section, the adding of IPv6 default routes would allow the clients to use the IPv6 connection.

To secure this new road onto the Internet we had to implement a second ruleset for IPv6. In this ruleset we implemented the same basic functionality as the IPv4 connection, minus the NAT service. This means that we open up for outgoing connections, and let their established counterparts back in the network. We also added rules to make hosts from the IPv6 Internet reach the specific hosts we are using to serve external services.

Just like the 6to4 transition mechanism, ISATAP routing has some special cases that should be treated with extra care in the firewall/router setup on the network. In our implementation we have the ISATAP router logically behind the firewall and this makes the firewall ISATAP unaware, all such ingress and egress checks will then have to done here. The firewall will only take care of the IPv6 part of the firewalling, while the ISATAP router handles the special cases added by ISATAP.

Table 14: ISATAP router rules

Rules	**Explanation**
Incoming	
Drop traffic to other than your own ISATAP subnet	It are clearly not for your clients
Encapsulate IPv6 packets and send IPv4 packets to the corresponding ISATAP host	According to standard
Outgoing	
Drop packets where the IPv4 source address in the encapsulation do not match the IPv6 source address	Only send well formed packets
Drop packets with reserved or otherwise restricted IPv4 addresses (Note that RFC1918 addresses are OK)	Impossible combinations
Drop packets going out to non global unicast IPv6 addresses	Non global unicast are not to be sent to the Internet

If the ISATAP router is following these rules, we have achieved that all ISATAP specific rules will be enforced and the outer firewall can regard all traffic coming in and out from the network as ”normal” IPv6 traffic and set up the rules based on that assumption.

7.3.4 General ISATAP considerations

When looking at the LAN/WAN/Firewall changes done here you can start to wonder why someone would go through all the trouble of setting up ISATAP if you have got an IPv6 connection coming in to your network. This is a valid concern that implementers should ask themselves. How much will the deployment of ISATAP cost versus the cost of just implementing native IPv6 throughout the network? There are perfectly good reasons to make the choice of using ISATAP for this scenario.

Implementing in smaller sections As we saw in the LAN section, there is no problem implementing ISATAP in smaller steps and a few clients at a time. This can help administrators in doing smaller sections of the network first without enabling IPv6 specific configuration services on the network. You do not have to worry if hosts with IPv6 enabled network stacks are going get a connection before they should, you just enable the ISATAP client on the selected and do not let native IPv6 ”loose” on the network.

The usage of IPv6 unaware equipment You have equipment in your network that can not handle the deployment of native IPv6, and needs to be replaced for it to work. This could typically be networking equipment that routes in the IPv4 segment, but can not handle IPv6. With ISATAP, these hosts continue to route IPv4 as nothing had happened, but will effectively be handling your IPv6 traffic as well.

Easy transition to native IPv6 Since the ISATAP mechanism needs a client to function, the adding of a native connection will in many cases be turning off the ISATAP client and enable ”proper” IPv6 configuration mechanisms. The IPv6 prefix used will be the same, the hosts can get the same IP addresses if this is desirable and in many cases you do not have to do anything, Windows hosts disable ISATAP when a proper IPv6 connection is available. The Linux hosts will need some work as the `miredo` package must be removed.

If none of these reasons are applicable to your networking setup, it should be considered to do a full IPv6 native deployment instead of the ISATAP route. Sometime in the future network administrators have to move from ISATAP to native IPv6 regardless, and this will create another implementation job. With native IPv6 you get a more transparent network in the sense that we do not mix the two IP versions and your IPv6 connection will not be dependant of a functioning IPv4 infrastructure.

ISATAP also raises some questions for the maintainers of security systems applied in the network. Are these systems able to function on tunneled traffic. Can e.g. your IDS inspect protocol 41 packets and look through the IPv6 payload to do their job? In this case it is not enough that such systems are IPv6 aware, they need to be ISATAP aware to be able to understand the special packets running over their IPv4 links.

This dependence on IPv4 is also raising concerns from networking professionals. The usage of DNS to create the `potential router list` is in direct conflict with one of the fundamental ”rules” that network professionals try live by. RFC1958 [36] clearly states that we should avoid circular dependencies:

> ... routing must not depend on look-ups in the Domain Name System (DNS), since the updating of DNS servers depends on successful routing.

This is precisely what ISATAP does and are because of this met with scepticism from people advocating that we follow and should put great effort in abiding the basic principles of the Internet. In our case this means that if the DNS service is stopped, the clients will not be able to locate the ISATAP router and thus not being able to enter the IPv6 Internet, making IPv6, a layer 3 protocol, totally dependant on the information from the higher layered protocol of UDP/TCP.

7.4 Teredo

In this section we will sum up the findings from the experiment with adding Teredo to our baseline network. We will look at how the different places in the network is affected (LAN, WAN and Firewall) and end with some thoughts around how this transition mechanism affects the network and its operation.

7.4.1 LAN alterations

As we saw in the Teredo experiment we did not touch the internal workings of the IPv4 part of the LAN segment. Everything is done on the client itself. The implementation in terms of getting it to work is very simple. The protocol is designed to traverse through firewalls and NAT boxes with the help of ordinary UDP traffic. This is done very well and the Windows hosts are not even subject to configuration. They came with Teredo default enabled and had the interface up and configured the first time it booted, making it routable from the IPv6 Internet. The same thing was experienced on the Linux hosts, although they did not have it enabled by default they required the installation of a Teredo daemon called `miredo` and everything worked without the need to configure anything.

When examining the different Teredo clients in use on the different hosts we found differences between them that implementers have to be aware of before deciding to use Teredo in their networks. The Windows client is enabled by default and will get you a Teredo link if it is capable of it. The fact that it is enabled by default has made Microsoft restrict the usage of the client. Basically you can only use the Teredo interface when contacting IPv6 addresses directly, you will not be able to use AAAA records and unsolicited incoming connections are dropped. It can be said that they have sacrificed the usability of the client for security without adding an easy way of using the Teredo client as a ”normal” connection.

The Debian/Ubuntu communities has on the other hand chosen a different approach to the same problem. They have not enabled the client by default, but made it easy to add for system administrators at a later time. This gives the users the chance of using Teredo as a fully featured IPv6 link with all the pros and cons of the technology, but only if you make a specific decision to do so. It can said to do the opposite of Microsoft, sacrificing some security for usability.

These different modes of operation makes it difficult to use Teredo as a transition technology that should enable all clients access to the IPv6 Internet. In its current state, Teredo is working very well on both platforms, but as we saw in Section 6.3 it will need some creative alterations to be useful on Windows, and cannot be said to be stable for everyday use in this state.

Teredo traffic can be compared to the previously discussed ISATAP traffic in the sense that it

”hides” the IPv6 traffic. Where ISATAP traffic is easy to spot on the protocol 41 header, the Teredo traffic is more disguised. It does not have such easy-to-spot features, but look like ordinary IPv4 UDP packets. This means that services that are put in place to watch the network have to be Teredo aware and be able to spot and control this traffic.

7.4.2 WAN alterations

When talking about WAN changes in respect to the Teredo alterations done in the experiment, we do not mean the WAN side of the firewall as could be assumed. In this setup all the clients in the network can be said to have WAN connections as they all terminate their own IPv6 connection and are part of the IPv6 Internet. The external connection of the gateway is being used to connect to the IPv6 Internet, but is happily unaware that this is the case.

A consequence of every host having a direct link to the IPv6 Internet is the flow of internal traffic. Two hosts sitting in the same office that want to establish a IPv6 connection between them are not aware of each others presence and are forced to go via the external Teredo server/relays to send the traffic. This is of course a very expensive solution with regards to networking resources used. Not only will the connection have higher latency than normal but it will also use the companies Internet connection in both directions and add extra strain to the resource. All the traffic will also be on the open Internet and then being transferred over a much more insecure medium than the internal LAN.

In the nature of Teredo we see some oddities, the connection will use a long time get set up. The reason for this is explained in Section 4.2.5 and can be briefly explained as a result of the somewhat complicated procedure of setting up the connection through the NAT device. Once the connection has been established we do however see good latency and throughput.

7.4.3 Firewall alterations

The changes made to the firewall when implementing Teredo is very easy to sum up. It is not being altered in any way. It is totally ignorant of the fact that IPv6 connections are flowing through it via UDP packets. This is the big advantage of Teredo, but also the major objection from security people. The firewalling of Teredo connections has effectively been moved from the networks firewall to the clients themselves.

Every connection is being established from the inside and are allowed through the firewall, the corresponding then is in the established state and the default policy of the firewall will let these packets back in. This also goes past the NAT without any trace of the ”hidden” traffic.

There are ways of making the firewall aware of this traffic, Teredo awareness is something that should not be hard to implement. This will however create an extra workload for the firewall, when such packets are found it is up to the administrator if it should be allowed or not. The problem here is that there is no effective middle way, we can not control the traffic when we spot it, we only have the choice of allowing or denying. There are multiple possible ways of doing this:

Disabling outgoing UDP If the Teredo client is denied outgoing UDP connection it would not be able to establish the outgoing connections. For obvious reasons this would also disable all other traffic that uses UDP and maybe are desired in the network.

Disable port UDP/3544 The standard port for Teredo connections are the UDP port 3544. If this is disabled it should have the same effect as the disabling of UDP as a whole. This port is however possible to change and does not require much work to switch to another to make sure the connection is kept alive.

Decapsulate and look for IPv6 If we enable the firewall to look especially at UDP traffic and teach it to recognise encapsulated IPv6 traffic, we can make a rule that drops such packets. We can even look for source addresses on IPv6 in the Teredo prefix `2001::/32`. This would only hurt Teredo traffic, but would be very costly in terms of CPU usage on the firewall and even this will not be able to catch encrypted UDP traffic.

Without the proper mechanisms in place to manage such connections on our network, we will have hosts in our network that are directly connectable from the IPv6 Internet and as previously mentioned will have to firewall the connections themselves. This means that we suddenly have a firewall on each host in the network that are under the scrutiny of the entire IPv6 Internet and should as a consequence of this use strict rules for the connections. This is not necessarily a problem, but adds to the complexity of the network and makes the firewalling of the network much harder.

7.4.4 General Teredo considerations

Teredo as a transition mechanism is quite different from the previous two discussed in this chapter. It does not care about the internal traffic of the local network, but only focuses on the acquiring of IPv6 connectivity for the single host. It does not use the well established standard of protocol 41 tunneling, but have created a more elusive UDP tunneling technique.

As we saw in Section 4.2.5, the way Teredo obtains a global unicast address gives out very much information about the host and network that the client is on. It will reveal the official IPv4 address, the port used for connections, which Teredo server is in use and the NAT type that the client is behind. This information is not something all network administrators would like to be known to the Internet, but as long as Teredo is used this information will be in the logs of every server that the client connects to.

Just like 6to4, Teredo is dependant on resources on the public part of the Internet. Namely the Teredo relays that are actually handling most of the traffic flow. These relays are run by certain ISP's that are doing this to enable their customers to use Teredo with good performance. But if the usage would suddenly be multiplied, will these hosts be able to manage all the traffic or would they start to limit who is able to use the relays? Running such relays can be costly in terms of network usage and the continued operation can be the source of discussion by the service providers. The amount of AS currently advertising the Teredo prefix are not very high, only 11, as we can watch on the site bgpmon.net [103]. Ideally every ISP should run this service to enable their customers to use good performing Teredo services. As this costs money, and they observe that it is working at the moment, many ISP's are not taking the effort to implement such relays.

7.5 Summary

All the three mechanisms that we have examined in the experiments have a common goal in delivering IPv6 connectivity to the hosts on the network. The way they are going about this is however very different. All the different techniques have different scopes of operation. 6to4 can provide IPv6 addresses to almost an infinite number of clients with the help of one global unicast IPv4 address. ISATAP focuses on delivering a way of distributing IPv6 connectivity to the network as a whole, but will not give global connectivity itself. Teredo is on the diametrical opposite of 6to4, it has no intention of providing connectivity to other than the host, but will in the vast majority of cases get an IPv6 link from almost everywhere. We have summed up the main differences of the techniques in Table 15 to make a more easy-to-understand and quick way of seeing main aspects of them.

Table 15: Summary of tested transition mechanisms

	6to4	**ISATAP**	**Teredo**
NAT Traversal	No	No	Yes
Tunneling technique	proto 41	proto 41	UDP encapsulation
IPv4 requirement	Global unicast IPv4	No	No
IPv4 dependency	Yes and No	Yes	Yes
IPv6 prefix	2002::/16	N/A	2001::/32
Available IPv6 space	/48 pr IPv4	N/A	/128
Transition to native IPv6	Medium	Easy	Hard
Implementation	Hard	Medium	Easy
OS support	Good	Good	Good
Usage	WAN/LAN	LAN	Single host
Recommended for transition	Yes	Yes	No

From the table we can draw some conclusions on the different mechanisms. First and foremost we can under no circumstance recommend Teredo as a transition mechanism for entire networks. It will work fine for its intended purpose, namely giving IPv6 connectivity to single hosts from virtually anywhere but the security issues in controlling the traffic will make secure administration almost impossible. In many cases it should actually be processes set in motion that denies this traffic.

For smaller networks, the 6to4 mechanism would be the choice for the authors of this thesis. This is based on the setup in the experiment with the 6to4 router working on the edge of the network. This gives us one point of IPv6 termination and the firewall sitting after the 6to4 router can be configured to handle the IPv6 traffic as native and implement necessary security features. The internal network will also be fully dual stacked and as a result of this be ready when native IPv6 should be implemented one day.

In networks that have ageing equipment or otherwise equipment not able to handle native IPv6 internally, ISATAP is a good mechanism. It will also suit administrators that would like to implement IPv6 on a smaller part of the network giving them an easier way of controlling which clients that are made IPv6 enabled. They will however need to make sure that the external firewall and the ISATAP router is working together to secure the IPv6 connectivity. Internal

segregation will work fine as ISATAP does IPv6 in IPv4 tunneling and is affected by IPv4 network restrictions.

We can also see scenarios where the combination of 6to4 and ISATAP could be useful, 6to4 handles the external IPv6 connectivity like in our experiment. While ISATAP will take on the responsibility to deliver the internal connectivity. This would give a quick way of getting IPv6 to a selected set of internal hosts without having a native IPv6 connection from your ISP.

8 Conclusion

We will now give the answers we have found regarding the research question posed in Section 1.5. Regarding the promised analysis of different transitional techniques, we have with the combined effort of the work done in Section 3.2 and the further analysis in Chapter 4 found that the tunneling techniques are the ones most suited for today's scenario. Especially 6to4 and ISATAP, while Teredo can be a good alternative for getting "quick and dirty" access.

The planning and execution of the experiments showed the necessity of detailed planning. We applied a divide and conqueror tactic that enabled us to break down the network in smaller implementation tasks, subsequently giving us smaller steps to implement and check for errors. This approach was proven successful and can be recommended for usage in real world scenarios.

In Chapter 7 we enumerated the changes done to the baseline network, analysed their influence on the running network in respect to the areas defined in Section 1.5. The discussion around the alterations and the concluding summary of the chapter have showed that that the different techniques are affecting the network in different ways, 6to4 affects both WAN/LAN, ISATAP is more LAN based and Teredo does frightening things for the overall network.

Our work shows that the techniques available today should enable administrators of small and medium sized networks to deploy IPv6 in a way that does not jeopardise the functionality and security of the existing setup. Our work gives a proposed way of doing this implementation from planning to execution that administrators can benefit from using in their organisations.

Our work also clearly shows that if administrators choose not to deploy IPv6, they should at least learn about IPv6 and acquire the knowledge of the basic operation. The work regarding Teredo showed us how the protocol can drill holes in almost every firewall. This proves that IPv6 is something that administrators should be aware of and if control over such traffic is desirable, security functions that are IPv6 aware must be implemented.

9 Future work

While writing this thesis we have come across different sections that we feel need further study. The following is our proposals for areas that we think should be target for further scrutiny.

- Using the line of action proposed in this thesis on a real world running network. This to do a case study with very concrete limitations and problems, giving highly relevant experience and feedback for other implementers.
- In this thesis we have looked at implementation in smaller/medium sized networks, more practical analysis and/or case studies of larger more complex networks would be interesting as much of the literature is very theoretical.
- We have discussed the lack of interest from ISP and some research into why they are so reluctant in deploying new services to their customers would be enlightening. This could help us understand why they hold back and how we can give them a push in the right direction.
- While this thesis focus on the transitional techniques for business networks, a quantitative study on how such techniques could be used by ISPs would be interesting. How they can be used to get started and shed some light on which are applicable and which are not.
- The findings regarding the Teredo protocol is something that should be investigated further. Work on how to effictively mitigate the usage if this protocol and work on the detection of usage would be enlightening.

Bibliography

[1] Deering, S. & Hinden, R. December 1998. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), http://www.ietf.org/rfc/rfc2460.txt. Updated by RFCs 5095, 5722.

[2] Hagen, S. 2006. *IPv6 Essentials*. O'Reilly Media, Inc., second edition.

[3] Malone, D. & Murphy, N. R. 2005. *IPv6 Network Administration*. O'Reilly Media, Inc.

[4] Hogg, S. & Vyncke, E. 2008. *IPv6 Security*. Cisco Press.

[5] Radhakrishnan, R., Jamil, M., Mehfuz, S., & Moinuddin, M. June 2007. Security issues in IPv6. *International Conference on Networking and Services (ICNS '07)*, 110–110.

[6] Caicedo, C. E., Joshi, J. B., & Tuladhar, S. R. February 2009. IPv6 Security Challenges. *Computer*, 42(2), 36–42.

[7] Waddington, D. June 2002. Realizing the transition to IPv6. *IEEE Communications Magazine*, 40(6), 138–148.

[8] Govil, J., Govil, J., Kaur, N., & Kaur, H. April 2008. An examination of IPv4 and IPv6 networks : Constraints and various transition mechanisms. *IEEE SoutheastCon 2008*, 178–185.

[9] Hoagland, J. 2006. The teredo protocol: Tunneling past network security and other security implications. *Symantec Response whitepaper, November*.

[10] Savola, P. & Patel, C. December 2004. Security Considerations for 6to4. RFC 3964 (Informational), http://www.ietf.org/rfc/rfc3964.txt.

[11] Bound, J., Pouffary, Y., Klynsma, S., Chown, T., & Green, D. April 2007. IPv6 Enterprise Network Analysis - IP Layer 3 Focus. RFC 4852 (Informational), http://www.ietf.org/rfc/rfc4852.txt.

[12] Carlisle, H. & Bailey, B. 2008. Enabling IPv6 within a campus network. In *Proceedings of the 46th Annual Southeast Regional Conference on XX*, 454–457. ACM.

[13] Yan-ge, C., Zi-yi, H., & Shui-mu, T. 2009. IPv6 Network Comprehensive Deployment on Campus Network. *2009 International Conference on Environmental Science and Information Application Technology*, 631–634.

[14] January 2010. Establishments, by size groups and county. 1st January 2010. http://www.ssb.no/english/subjects/10/01/bedrifter_en/tab-2010-01-29-02-en.html. Last visited: 10.06.2010.

[15] Wikipedia - Digital Divide. http://en.wikipedia.org/wiki/Digital_divide. Last visited: 26.05.2010.

[16] April 2010. IPv4's Last Day: What Will Happen When There Is Only IPv6? http://www.enterprisenetworkingplanet.com/news/article.php/3878391. Last visited: 25.05.2010.

[17] IEEE-USA. 2009. Next Generation Internet: IPv4 Address Exhaustion, Mitigation Strategies and Implications for the U.S.

[18] Knuth, D. E. 1998. *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[19] Leedy, P. D. & Ormrod, J. E. 2009. *Practical Research: Planning and Design, 9th Edition*. Prentice Hall.

[20] Hinden, R. & Deering, S. February 2006. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard) http://www.ietf.org/rfc/rfc4291.txt.

[21] Unicast - From Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Unicast. Last visited: 10.06.2010.

[22] Multicast - From Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Multicast. Last visited: 10.06.2010.

[23] Anycast - From Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Anycast. Last visited: 10.06.2010.

[24] Hinden, R. & Haberman, B. October 2005. Unique Local IPv6 Unicast Addresses. RFC 4193 (Proposed Standard) http://www.ietf.org/rfc/rfc4193.txt.

[25] Internet Assigned Numbers Authority (IANA). Number resources. http://www.iana.org/numbers/. Last visited: 31.05.2010.

[26] Droms, R. March 1997. Dynamic Host Configuration Protocol. RFC 2131 (Draft Standard) http://www.ietf.org/rfc/rfc2131.txt. Updated by RFCs 3396, 4361, 5494.

[27] Thomson, S., Narten, T., & Jinmei, T. September 2007. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard) http://www.ietf.org/rfc/rfc4862.txt.

[28] Institute of Electrical and Electronics Engineers (IEEE). Guidelines for 64-bit global identifier (eui-64). http://standards.ieee.org/regauth/oui/tutorials/EUI64.html. Last visited: 31.05.2010.

[29] August 2008. EUI-64 in IPv6. http://packetlife.net/blog/2008/aug/04/eui-64-ipv6/. Last visited: 10.06.2010.

[30] Narten, T., Draves, R., & Krishnan, S. September 2007. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (Draft Standard), http://www.ietf.org/rfc/rfc4941.txt.

[31] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., & Carney, M. July 2003. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315 (Proposed Standard), http://www.ietf.org/rfc/rfc3315.txt. Updated by RFCs 4361, 5494.

[32] 2008. Internet users per 100 inhabitants. http://www.itu.int/ITU-D/ict/statistics/ict/graphs/internet.jpg. Last visited: 10.06.2010.

[33] January 2010. To 4,294,967,296 and Beyond - Under 10% of IPv4 Space Remains: Adoption of IPv6 Is Essential. Web, http://www.icann.org/en/announcements/announcement-29jan10-en.htm. Last visited: 31.05.2010.

[34] Saltzer, J. H., Reed, D. P., & Clark, D. D. November 1984. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4), 277–288.

[35] Clark, D. 1988. The design philosophy of the darpa internet protocols. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, 106–114, New York, NY, USA. ACM.

[36] Carpenter, B. June 1996. Architectural Principles of the Internet. RFC 1958 (Informational), http://www.ietf.org/rfc/rfc1958.txt. Updated by RFC 3439.

[37] Bush, R. & Meyer, D. December 2002. Some Internet Architectural Guidelines and Philosophy. RFC 3439 (Informational), http://www.ietf.org/rfc/rfc3439.txt.

[38] Srisuresh, P. & Ford, B. February 2010. Unintended Consequences of NAT Deployments with Overlapping Address Space. RFC 5684 (Informational), http://www.ietf.org/rfc/rfc5684.txt.

[39] Wilson, P., Plzak, R., & Pawlik, A. February 2005. Ipv6 address space management. http://www.ripe.net/ripe/docs/ipv6-sparse.html. Last visited: 31.05.2010.

[40] Carr, B., Sury, O., Martinez, J. P., Davidson, A., Evans, R., Yilmaz, F., & Wijte, I. September 2009. Ipv6 address allocation and assignment policy. http://www.ripe.net/ripe/docs/ipv6policy.html. Last visited: 29.05.2010.

[41] Postel, J. September 1981. Internet Protocol. RFC 791 (Standard), http://www.ietf.org/rfc/rfc791.txt. Updated by RFC 1349.

[42] Atkinson, R. August 1995. Security Architecture for the Internet Protocol. RFC 1825 (Proposed Standard), http://www.ietf.org/rfc/rfc1825.txt. Obsoleted by RFC 2401.

[43] Kent, S. & Atkinson, R. November 1998. Security Architecture for the Internet Protocol. RFC 2401 (Proposed Standard), http://www.ietf.org/rfc/rfc2401.txt. Obsoleted by RFC 4301, updated by RFC 3168.

[44] Kent, S. & Seo, K. December 2005. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), http://www.ietf.org/rfc/rfc4301.txt.

[45] Curran, J. July 2008. An Internet Transition Plan. RFC 5211 (Informational), http://www.ietf.org/rfc/rfc5211.txt.

[46] Leng, X., Bi, J., & Zhang, M. 2006. Study on High Performance IPv4/IPv6 Transition and Access Service. *Lecture Notes in Computer Science*, 4330, 183.

[47] Bound, J. June 2005. IPv6 Enterprise Network Scenarios. RFC 4057 (Informational), http://www.ietf.org/rfc/rfc4057.txt.

[48] Sailan, M., Hassan, R., & Patel, A. 2009. A Comparative Review of IPv4 and IPv6 for Research Test Bed. *ftsm.ukm.my*, (August), 427–433.

[49] Ipv6 ready program. http://www.ipv6ready.org. Last visited: 12.06.2010.

[50] Nordmark, E. & Gilligan, R. October 2005. Basic Transition Mechanisms for IPv6 Hosts and Routers. RFC 4213 (Proposed Standard), http://www.ietf.org/rfc/rfc4213.txt.

[51] Carpenter, B. & Moore, K. February 2001. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056 (Proposed Standard), http://www.ietf.org/rfc/rfc3056.txt.

[52] Templin, F., Gleeson, T., & Thaler, D. March 2008. Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). RFC 5214 (Informational), http://www.ietf.org/rfc/rfc5214.txt.

[53] Visoottiviseth, V. & Bureenok, N. March 2008. Performance Comparison of ISATAP Implementations on FreeBSD, RedHat, and Windows 2003. *22nd International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008)*, 547–552.

[54] Huitema, C. February 2006. Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs). RFC 4380 (Proposed Standard), http://www.ietf.org/rfc/rfc4380.txt.

[55] Huang, S.-m., Wu, Q., & Lin, Y.-b. 2005. Tunneling IPv6 through NAT with Teredo Mechanism. *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*, 813–818.

[56] Tsirtsis, G. & Srisuresh, P. February 2000. Network Address Translation - Protocol Translation (NAT-PT). RFC 2766 (Historic), http://www.ietf.org/rfc/rfc2766.txt. Obsoleted by RFC 4966, updated by RFC 3152.

[57] Thomson, S., Huitema, C., Ksinant, V., & Souissi, M. October 2003. DNS Extensions to Support IP Version 6. RFC 3596 (Draft Standard), http://www.ietf.org/rfc/rfc3596.txt.

[58] Vazao, T., Raposo, L., & Santos, J. 2004. Migration to the New Internet-Supporting Inter Operability Between IPv4 and IPv6 Networks. *Lecture notes in computer science*, 678–687.

[59] Waddington, D. G., Chang, F., Viswanathan, R., & Yao, B. July 2003. Topology discovery for public IPv6 networks. *ACM SIGCOMM Computer Communication Review*, 33(3), 59.

[60] Aoun, C. & Davies, E. July 2007. Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status. RFC 4966 (Informational), http://www.ietf.org/rfc/rfc4966.txt.

[61] Xia, H., Bound, J., & Pouffary, Y. 1994. Evaluating DSTM–an IPv6 Transition Mechanism for IPv6 Dominant Networks. *Citeseer*.

[62] Bound, J., Toutain, L., & Richier, J. 2005. Dual Stack IPv6 Dominant Transition Mechanism (DSTM). http://tools.ietf.org/html/draft-bound-dstm-exp-04.

[63] Kong, I., Lee, J., & Lee, J. 2003. Design and Implementation of IPv6-IPv4 Protocol Translation System Using Dynamic IP Address. *Lecture Notes in Computer Science*, 496–506.

[64] Law, Y.-N., Lai, M.-C., Tan, W. L., & Lau, W. C. May 2008. Empirical Performance of IPv6 vs. IPv4 under a Dual-Stack Environment. *2008 IEEE International Conference on Communications*, 5924–5929.

[65] Wang, Y., Ye, S., & Li, X. 2005. Understanding current IPv6 performance: a measurement study. In *10th IEEE Symposium on Computer Communications*. Citeseer.

[66] Hirorai, R. & Yoshifuji, H. 2006. Problems on IPv4-IPv6 network transition. In *Applications and the Internet Workshops, 2006. SAINT Workshops 2006. International Symposium on*, 5.

[67] Davies, E., Krishnan, S., & Savola, P. September 2007. IPv6 Transition/Co-existence Security Considerations. RFC 4942 (Informational), http://www.ietf.org/rfc/rfc4942.txt.

[68] Lancaster, T. 2006. IPv6 \& IPv4 Threat Review with Dual-Stack Considerations. *Online*, (May), 1–8.

[69] Lai, Y., Jiang, G., Li, J., & Yang, Z. 2009. Design and Implementation of Distributed Firewall System for IPv6. *2009 International Conference on Communication Software and Networks*, 428–432.

[70] IPv6 Operating Systems. http://ipv6int.net/systems/index.html. Last visited: 11.06.2010.

[71] Microsoft Internet Protocol Version 6 (IPv6). http://www.microsoft.com/ipv6. Last visited: 11.06.2010.

[72] Frankel, S., Graveman, R., & Pearce, J. 2010. Guidelines for the Secure Deployment of IPv6 (Draft) Recommendations of the National Institute of Standards and Technology. *Nist Special Publication*, 119.

[73] Internet Assigned Numbers Authority (IANA). May 2010. Protocol numbers. http://www.iana.org/assignments/protocol-numbers/. Last visited: 31.05.2010.

[74] Gilligan, R. & Nordmark, E. August 2000. Transition Mechanisms for IPv6 Hosts and Routers. RFC 2893 (Proposed Standard), http://www.ietf.org/rfc/rfc2893.txt. Obsoleted by RFC 4213.

[75] Crawford, M., Narten, T., & Thomas, S. December 1998. Transmission of IPv6 Packets over Token Ring Networks. RFC 2470 (Proposed Standard), http://www.ietf.org/rfc/rfc2470.txt.

[76] Crawford, M. December 1998. Transmission of IPv6 Packets over Ethernet Networks. RFC 2464 (Proposed Standard), http://www.ietf.org/rfc/rfc2464.txt.

[77] Carpenter, B. & Jung, C. March 1999. Transmission of IPv6 over IPv4 Domains without Explicit Tunnels. RFC 2529 (Proposed Standard), http://www.ietf.org/rfc/rfc2529.txt.

[78] SixXS - IPv6 Deployment and Tunnel Broker. http://www.sixxs.net. Last visited: 31.05.2010.

[79] Hurricane Electric - Internet Backbone and Colocation Provider. http://www.he.net. Last visited: 31.05.2010.

[80] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., & Lear, E. February 1996. Address Allocation for Private Internets. RFC 1918 (Best Current Practice), http://www.ietf.org/rfc/rfc1918.txt.

[81] Blanchet, M. April 2008. Special-Use IPv6 Addresses. RFC 5156 (Informational), http://www.ietf.org/rfc/rfc5156.txt.

[82] Wireshark - the open-source packet analyser. http://www.wireshark.org/. Last visited: 31.05.2010.

[83] Nordmark, E. February 2000. Stateless IP/ICMP Translation Algorithm (SIIT). RFC 2765 (Proposed Standard), http://www.ietf.org/rfc/rfc2765.txt.

[84] Srisuresh, P., Tsirtsis, G., Akkiraju, P., & Heffernan, A. September 1999. DNS extensions to Network Address Translators (DNS_ALG). RFC 2694 (Informational), http://www.ietf.org/rfc/rfc2694.txt.

[85] Egevang, K. & Francis, P. May 1994. The IP Network Address Translator (NAT), http://www.ietf.org/rfc/rfc1631.txt. RFC 1631 (Informational). Obsoleted by RFC 3022.

[86] Srisuresh, P. & Egevang, K. January 2001. Traditional IP Network Address Translator (Traditional NAT), http://www.ietf.org/rfc/rfc3022.txt. RFC 3022 (Informational).

[87] Li, X., Bao, C., & Baker, F. 2010. Framework for IPv4/IPv6 Translation. draft-ietf-behave-v6v4-framework-08, http://tools.ietf.org/html/draft-ietf-behave-v6v4-framework-08.

[88] Li, X., Bao, C., & Baker, F. 2010. IP/ICMP Translation Algorithm. draft-ietf-behave-v6v4-xlate-12, http://tools.ietf.org/html/draft-ietf-behave-v6v4-xlate-12. Last visited: 31.05.2010.

[89] Hagino, J. & Yamamoto, K. June 2001. An IPv6-to-IPv4 Transport Relay Translator. RFC 3142 (Informational), http://www.ietf.org/rfc/rfc3142.txt.

[90] Tsuchiya, K., Higuchi, H., & Atarashi, Y. February 2000. Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS). RFC 2767 (Informational), http://www.ietf.org/rfc/rfc2767.txt.

[91] Lee, S., Shin, M.-K., Kim, Y.-J., Nordmark, E., & Durand, A. October 2002. Dual Stack Hosts Using "Bump-in-the-API" (BIA). RFC 3338 (Experimental), http://www.ietf.org/rfc/rfc3338.txt.

[92] VMware virtualisation software. http://www.vmware.com/. Last visited: 12.06.2010.

[93] Massar, J. July 2004. Ayiya: Anything in anything. http://unfix.org/~jeroen/archive/drafts/draft-massar-v6ops-ayiya-02.txt. Last visited: 31.05.2010.

[94] m0n0wall embedded firewall. http://www.m0n0.ch. Last visited: 31.05.2010.

[95] Debian - the free operating system. http://www.debian.org. Last visited: 31.05.2010.

[96] The Apache web server project. http://httpd.apache.org/. Last visited: 31.05.2010.

[97] OpenSSH - Keeping Your Communiques Secret. http://www.openssh.com/. Last visited: 31.05.2010.

[98] Microsoft Server 2008. http://www.microsoft.com/windowsserver2008/. Last visited: 31.05.2010.

[99] Ubuntu by Canonical. http://www.ubuntu.com/. Last visited: 31.05.2010.

[100] Microsoft Windows 7. http://www.microsoft.com/windows/windows-7/. Last visited: 31.05.2010.

[101] DynDNS - Dynamic Network Services. http://www.dyndns.com/. Last visited: 01.06.2010.

[102] BGPmon.net - a BGP monitoring and analyzer tool, 6to4 relays. http://bgpmon.net/6to4.php. Last visited: 31.05.2010.

[103] BGPmon.net - a BGP monitoring and analyzer tool, Teredo relays. http://bgpmon.net/teredo.php. Last visited: 31.05.2010.

A Configuration files

This appendix contains different configuration files that have been used throughout the thesis.

A.1 m0n0wall

The m0n0wall firewall distribution are quite unique with the fact that all the configuration done is saved in .xml-files that you can export for backup needs. This section contains the .xml-files for the different configurations described in the thesis.

A.1.1 m0n0wall IPv4 only

```
<?xml version="1.0"?>
<m0n0wall>
        <version>1.8</version>
        <lastchange>1271777265</lastchange>
        <system>
                <hostname>m0n0wall</hostname>
                <domain>example.com</domain>
                <dnsallowoverride/>
                <username>admin</username>
                <password>$1$VmblDx8x$B0vMSlFT9Li6CBVxEwVjV0</password>
                <timezone>Europe/Oslo</timezone>
                <time-update-interval>300</time-update-interval>
                <timeservers>1.m0n0wall.pool.ntp.org</timeservers>
                <webgui>
                        <protocol>https</protocol>
                        <port/>
                        <certificate>REMOVED CERTIFICATE</certificate>
                        <private-key>REMOVED KEY</private-key>
                </webgui>
        </system>
        <interfaces>
                <lan>
                        <if>em1</if>
                        <ipaddr>192.168.1.1</ipaddr>
                        <subnet>24</subnet>
                        <media/>
                        <mediaopt/>
                </lan>
                <wan>
                        <if>em0</if>
                        <ipaddr>dhcp</ipaddr>
                        <subnet/>
                        <gateway/>
                        <blockpriv/>
                        <dhcphostname/>
                        <media/>
```

```
                <mediaopt/>
        </wan>
</interfaces>
<staticroutes/>
<pppoe>
        <username/>
        <password/>
        <provider/>
</pppoe>
<pptp>
        <username/>
        <password/>
        <local/>
        <subnet/>
        <remote/>
</pptp>
<dyndns>
        <type>dyndns</type>
        <username/>
        <password/>
        <host/>
        <mx/>
        <server/>
        <port/>
</dyndns>
<dnsupdate/>
<dhcpd>
        <lan>
                <enable/>
                <range>
                        <from>192.168.1.100</from>
                        <to>192.168.1.199</to>
                </range>
        </lan>
</dhcpd>
<pptpd>
        <mode/>
        <nunits>16</nunits>
        <redir/>
        <localip/>
        <remoteip/>
</pptpd>
<dnsmasq>
        <enable/>
        <regdhcp/>
</dnsmasq>
<snmpd>
        <syslocation/>
        <syscontact/>
        <rocommunity>public</rocommunity>
</snmpd>
<diag/>
<bridge/>
```

```
<syslog>
        <reverse/>
        <nentries>50</nentries>
        <remoteserver/>
        <remoteport/>
</syslog>
<nat>
        <rule>
                <protocol>tcp</protocol>
                <external-port>22</external-port>
                <target>192.168.1.196</target>
                <local-port>22</local-port>
                <interface>wan</interface>
                <descr/>
        </rule>
        <rule>
                <protocol>tcp</protocol>
                <external-port>80</external-port>
                <target>192.168.1.196</target>
                <local-port>80</local-port>
                <interface>wan</interface>
                <descr/>
        </rule>
        <rule>
                <protocol>tcp</protocol>
                <external-port>3389</external-port>
                <target>192.168.1.193</target>
                <local-port>3389</local-port>
                <interface>wan</interface>
                <descr>Remote Desktop</descr>
        </rule>
</nat>
<filter>
        <rule>
                <interface>wan</interface>
                <protocol>tcp</protocol>
                <source>
                        <any/>
                </source>
                <destination>
                        <address>192.168.1.196</address>
                        <port>22</port>
                </destination>
                <descr>NAT </descr>
        </rule>
        <rule>
                <interface>wan</interface>
                <protocol>tcp</protocol>
                <source>
                        <any/>
                </source>
                <destination>
                        <address>192.168.1.196</address>
```

```
                        <port>80</port>
                    </destination>
                    <descr>NAT </descr>
                </rule>
                <rule>
                    <type>pass</type>
                    <descr>Default LAN –&gt; any</descr>
                    <interface>lan</interface>
                    <source>
                        <network>lan</network>
                    </source>
                    <destination>
                        <any/>
                    </destination>
                </rule>
                <rule>
                    <type>pass</type>
                    <descr>Default IPsec VPN</descr>
                    <interface>ipsec</interface>
                    <source>
                        <any/>
                    </source>
                    <destination>
                        <any/>
                    </destination>
                </rule>
                <rule>
                    <interface>wan</interface>
                    <protocol>tcp</protocol>
                    <source>
                        <any/>
                    </source>
                    <destination>
                        <address>192.168.1.193</address>
                        <port>3389</port>
                    </destination>
                    <descr>NAT Remote Desktop</descr>
                </rule>
        </filter>
        <shaper/>
        <ipsec/>
        <aliases/>
        <proxyarp/>
        <wol/>
</m0n0wall>
```

A.1.2 m0n0wall IPv4 and IPv6

```
<?xml version="1.0"?>
<m0n0wall>
        <version>1.8</version>
        <lastchange>1271853623</lastchange>
        <system>
                <hostname>m0n0wall</hostname>
```

```
        <domain>example.com</domain>
        <dnsallowoverride/>
        <username>admin</username>
        <password>$1$VmblDx8x$B0vMSlFT9Li6CBVxEwVjV0</password>
        <timezone>Europe/Oslo</timezone>
        <time-update-interval>300</time-update-interval>
        <timeservers>1.m0n0wall.pool.ntp.org</timeservers>
        <webgui>
                <protocol>https</protocol>
                <port/>
                <certificate>REMOVED CERTIFICATE</certificate>
                <private-key>REMOVED KEY</private-key>
        </webgui>
        <harddiskstandby/>
        <enableipv6/>
</system>
<interfaces>
        <lan>
                <if>em1</if>
                <ipaddr>192.168.1.1</ipaddr>
                <subnet>24</subnet>
                <media/>
                <mediaopt/>
                <ipaddr6>2001:16d8:ee71::1</ipaddr6>
                <subnet6>64</subnet6>
        </lan>
        <wan>
                <if>em0</if>
                <blockpriv/>
                <media/>
                <mediaopt/>
                <spoofmac/>
                <aiccu>
                        <username>HDL1-SIXXS</username>
                        <password>REMOVED PASSWORD</password>
                        <tunnelid>T25221</tunnelid>
                        <ayiya/>
                </aiccu>
                <ipaddr>dhcp</ipaddr>
                <dhcphostname/>
                <ipv6ra/>
                <ipaddr6>aiccu</ipaddr6>
        </wan>
</interfaces>
<staticroutes/>
<pppoe/>
<pptp/>
<dyndns>
        <type>dyndns</type>
        <username/>
        <password/>
        <host/>
        <mx/>
```

```
        <server/>
        <port/>
</dyndns>
<dnsupdate/>
<dhcpd>
        <lan>
                <enable/>
                <range>
                        <from>192.168.1.100</from>
                        <to>192.168.1.199</to>
                </range>
        </lan>
</dhcpd>
<pptpd>
        <mode/>
        <nunits>16</nunits>
        <redir/>
        <localip/>
        <remoteip/>
</pptpd>
<dnsmasq>
        <enable/>
        <regdhcp/>
</dnsmasq>
<snmpd>
        <syslocation/>
        <syscontact/>
        <rocommunity>public</rocommunity>
</snmpd>
<diag/>
<bridge/>
<syslog>
        <reverse/>
        <nentries>50</nentries>
        <remoteserver/>
        <remoteport/>
</syslog>
<nat>
        <rule>
                <protocol>tcp</protocol>
                <external-port>22</external-port>
                <target>192.168.1.196</target>
                <local-port>22</local-port>
                <interface>wan</interface>
                <descr/>
        </rule>
        <rule>
                <protocol>tcp</protocol>
                <external-port>80</external-port>
                <target>192.168.1.196</target>
                <local-port>80</local-port>
                <interface>wan</interface>
                <descr/>
```

```
        </rule>
        <rule>
                <protocol>tcp</protocol>
                <external-port>3389</external-port>
                <target>192.168.1.193</target>
                <local-port>3389</local-port>
                <interface>wan</interface>
                <descr>Remote Desktop</descr>
        </rule>
        <portrange-low/>
        <portrange-high/>
</nat>
<filter>
        <rule>
                <interface>wan</interface>
                <protocol>tcp</protocol>
                <source>
                        <any/>
                </source>
                <destination>
                        <address>192.168.1.196</address>
                        <port>22</port>
                </destination>
                <descr>NAT </descr>
        </rule>
        <rule>
                <interface>wan</interface>
                <protocol>tcp</protocol>
                <source>
                        <any/>
                </source>
                <destination>
                        <address>192.168.1.196</address>
                        <port>80</port>
                </destination>
                <descr>NAT </descr>
        </rule>
        <rule>
                <type>pass</type>
                <descr>Default LAN -&gt; any</descr>
                <interface>lan</interface>
                <source>
                        <network>lan</network>
                </source>
                <destination>
                        <any/>
                </destination>
        </rule>
        <rule>
                <type>pass</type>
                <descr>Default IPsec VPN</descr>
                <interface>ipsec</interface>
                <source>
```

```
                        <any/>
                    </source>
                    <destination>
                        <any/>
                    </destination>
                </rule>
                <rule>
                    <interface>wan</interface>
                    <protocol>tcp</protocol>
                    <source>
                        <any/>
                    </source>
                    <destination>
                        <address>192.168.1.193</address>
                        <port>3389</port>
                    </destination>
                    <descr>NAT Remote Desktop</descr>
                </rule>
                <tcpidletimeout/>
                <rule6>
                    <type>pass</type>
                    <interface>lan</interface>
                    <source>
                        <network>lan</network>
                    </source>
                    <destination>
                        <any/>
                    </destination>
                    <descr/>
                </rule6>
        </filter>
        <shaper/>
        <ipsec>
                <dns-interval/>
        </ipsec>
        <aliases/>
        <proxyarp/>
        <wol/>
</m0n0wall>
```

A.1.3 m0n0wall 6to4

```
<?xml version="1.0"?>
<m0n0wall>
        <version>1.8</version>
        <lastchange>1271777265</lastchange>
        <system>
                <hostname>m0n0wall</hostname>
                <domain>example.com</domain>
                <dnsallowoverride/>
                <username>admin</username>
                <password>$1$VmblDx8x$B0vMSlFT9Li6CBVxEwVjV0</password>
                <timezone>Europe/Oslo</timezone>
                <time-update-interval>300</time-update-interval>
```

```
        <timeservers>1.m0n0wall.pool.ntp.org</timeservers>
        <webgui>
                <protocol>https</protocol>
                <port/>
    <certificate>REMOVED CERTIFICATE</certificate>
                <private-key>REMOVED KEY</private-key>
        </webgui>
        <harddiskstandby/>
        <enableipv6/>
</system>
<interfaces>
        <lan>
                <if>em1</if>
                <ipaddr>192.168.1.1</ipaddr>
                <subnet>24</subnet>
                <media/>
                <mediaopt/>
                <ipaddr6>2002:8027:50aa:1::1</ipaddr6>
                <ipv6ra/>
                <ipv6rao/>
                <subnet6>64</subnet6>
        </lan>
        <wan>
                <if>em0</if>
                <blockpriv/>
                <media/>
                <mediaopt/>
                <spoofmac/>
                <ipaddr>dhcp</ipaddr>
                <dhcphostname/>
                <ipaddr6>6to4</ipaddr6>
        </wan>
</interfaces>
<staticroutes/>
<pppoe/>
<pptp/>
<dyndns>
        <type>dyndns</type>
        <username/>
        <password/>
        <host/>
        <mx/>
        <server/>
        <port/>
</dyndns>
<dnsupdate/>
<dhcpd>
        <lan>
                <enable/>
                <range>
                        <from>192.168.1.100</from>
                        <to>192.168.1.199</to>
                </range>
```

```
                <defaultleasetime/>
                <maxleasetime/>
                <next-server/>
                <filename/>
                <v6range>
                        <from>2002:8027:50aa:1::100</from>
                        <to>2002:8027:50aa:1::150</to>
                </v6range>
                <v6defaultleasetime>7200</v6defaultleasetime>
                <v6maxleasetime>86400</v6maxleasetime>
                <v6enable/>
        </lan>
</dhcpd>
<pptpd>
        <mode/>
        <nunits>16</nunits>
        <redir/>
        <localip/>
        <remoteip/>
</pptpd>
<dnsmasq>
        <enable/>
        <regdhcp/>
        <hosts>
                <host>debian-server</host>
                <domain>example.com</domain>
                <ip>2002:8027:50aa:1::10</ip>
                <descr>IPv6 Debian</descr>
        </hosts>
        <hosts>
                <host>m0n0wall</host>
                <domain>example.com</domain>
                <ip>2002:8027:50aa:1::1</ip>
                <descr>IPv6 m0n0wall</descr>
        </hosts>
        <hosts>
                <host>win-server</host>
                <domain>example.com</domain>
                <ip>2002:8027:50aa:1::20</ip>
                <descr>IPv6 Windows server</descr>
        </hosts>
</dnsmasq>
<snmpd>
        <syslocation/>
        <syscontact/>
        <rocommunity>public</rocommunity>
</snmpd>
<diag/>
<bridge/>
<syslog>
        <reverse/>
        <nentries>50</nentries>
        <remoteserver/>
```

```
        <remoteport/>
</syslog>
<nat>
        <rule>
                <protocol>tcp</protocol>
                <external-port>22</external-port>
                <target>192.168.1.210</target>
                <local-port>22</local-port>
                <interface>wan</interface>
                <descr/>
        </rule>
        <rule>
                <protocol>tcp</protocol>
                <external-port>80</external-port>
                <target>192.168.1.210</target>
                <local-port>80</local-port>
                <interface>wan</interface>
                <descr/>
        </rule>
        <rule>
                <protocol>tcp</protocol>
                <external-port>3389</external-port>
                <target>192.168.1.220</target>
                <local-port>3389</local-port>
                <interface>wan</interface>
                <descr>Remote Desktop</descr>
        </rule>
        <portrange-low/>
        <portrange-high/>
</nat>
<filter>
        <rule>
                <type>pass</type>
                <interface>wan</interface>
                <protocol>tcp</protocol>
                <source>
                        <any/>
                </source>
                <destination>
                        <address>192.168.1.210</address>
                        <port>22</port>
                </destination>
                <descr>NAT </descr>
        </rule>
        <rule>
                <type>pass</type>
                <interface>wan</interface>
                <protocol>tcp</protocol>
                <source>
                        <any/>
                </source>
                <destination>
                        <address>192.168.1.210</address>
```

```
                <port>80</port>
        </destination>
        <descr>NAT </descr>
</rule>
<rule>
        <type>pass</type>
        <interface>wan</interface>
        <protocol>tcp</protocol>
        <source>
                <any/>
        </source>
        <destination>
                <address>192.168.1.220</address>
                <port>3389</port>
        </destination>
        <descr>NAT Remote Desktop</descr>
</rule>
<rule>
        <type>pass</type>
        <descr>Default LAN -&gt; any</descr>
        <interface>lan</interface>
        <source>
                <network>lan</network>
        </source>
        <destination>
                <any/>
        </destination>
</rule>
<rule>
        <type>pass</type>
        <descr>Default IPsec VPN</descr>
        <interface>ipsec</interface>
        <source>
                <any/>
        </source>
        <destination>
                <any/>
        </destination>
</rule>
<tcpidletimeout/>
<rule6>
        <type>pass</type>
        <interface>wan</interface>
        <protocol>tcp</protocol>
        <source>
                <any/>
        </source>
        <destination>
                <address>2002:8027:50aa:1::10</address>
                <port>22</port>
        </destination>
        <descr>SSH IPv6</descr>
</rule6>
```

```
                <rule6>
                        <type>pass</type>
                        <interface>wan</interface>
                        <protocol>tcp</protocol>
                        <source>
                                <any/>
                        </source>
                        <destination>
                                <address>2002:8027:50aa:1::10</address>
                                <port>80</port>
                        </destination>
                        <descr>IPv6 Web</descr>
                </rule6>
                <rule6>
                        <type>pass</type>
                        <interface>wan</interface>
                        <protocol>tcp</protocol>
                        <source>
                                <any/>
                        </source>
                        <destination>
                                <address>2002:8027:50aa:1::20</address>
                                <port>3389</port>
                        </destination>
                        <descr>IPv6 Remote desktop</descr>
                </rule6>
                <rule6>
                        <type>pass</type>
                        <interface>lan</interface>
                        <source>
                                <address>2002:8027:50aa::/48</address>
                        </source>
                        <destination>
                                <any/>
                        </destination>
                        <descr/>
                </rule6>
        </filter>
        <shaper/>
        <ipsec>
                <dns-interval/>
        </ipsec>
        <aliases/>
        <proxyarp/>
        <wol/>
</m0n0wall>
```

A.1.4 m0n0wall ISATAP

```
<?xml version="1.0"?>
<m0n0wall>
        <version>1.8</version>
        <lastchange>1273595041</lastchange>
        <system>
```

```
                <hostname>m0n0wall</hostname>
                <domain>example.com</domain>
                <dnsallowoverride/>
                <username>admin</username>
                <password>$1$VmblDx8x$B0vMSlFT9Li6CBVxEwVjV0</password>
                <timezone>Europe/Oslo</timezone>
                <time-update-interval>300</time-update-interval>
                <timeservers>1.m0n0wall.pool.ntp.org</timeservers>
                <webgui>
                        <protocol>https</protocol>
                        <port/>
                        <certificate>REMOVED CERTIFICATE</certificate>
                        <private-key>REMOVED KEY</private-key>
                        <mbmon>
                                <type>C</type>
                        </mbmon>
                </webgui>
                <harddiskstandby/>
                <enableipv6/>
        </system>
        <interfaces>
                <lan>
                        <if>em1</if>
                        <ipaddr>192.168.1.1</ipaddr>
                        <subnet>24</subnet>
                        <media/>
                        <mediaopt/>
                        <ipaddr6>2001:16d8:ee71:1::1</ipaddr6>
                        <subnet6>64</subnet6>
                </lan>
                <wan>
                        <if>em0</if>
                        <blockpriv/>
                        <media/>
                        <mediaopt/>
                        <spoofmac/>
                        <aiccu>
                                <username>HDL1-SIXXS</username>
                                <password>doWfVISl</password>
                                <tunnelid>T25221</tunnelid>
                                <ayiya/>
                        </aiccu>
                        <ipaddr>dhcp</ipaddr>
                        <dhcphostname/>
                        <ipv6ra/>
                        <ipaddr6>aiccu</ipaddr6>
                </wan>
        </interfaces>
        <staticroutes>
                <route6>
                        <interface>lan</interface>
                        <network>2001:16d8:ee71:2:0:0:0:0/64</network>
                        <gateway>2001:16d8:ee71:1::10</gateway>
```

```
                <descr>ISATAP router</descr>
        </route6>
</staticroutes>
<pppoe/>
<pptp/>
<dyndns>
        <type>dyndns</type>
        <username/>
        <password/>
        <host/>
        <mx/>
        <server/>
        <port/>
</dyndns>
<dnsupdate/>
<dhcpd>
        <lan>
                <enable/>
                <range>
                        <from>192.168.1.100</from>
                        <to>192.168.1.199</to>
                </range>
        </lan>
</dhcpd>
<pptpd>
        <mode/>
        <nunits>16</nunits>
        <redir/>
        <localip/>
        <remoteip/>
</pptpd>
<dnsmasq>
        <enable/>
        <regdhcp/>
        <hosts>
                <host>debian-server</host>
                <domain>example.com</domain>
                <ip>192.168.1.210</ip>
                <descr>IPv4 Debian</descr>
        </hosts>
        <hosts>
                <host>debian-server</host>
                <domain>example.com</domain>
                <ip>2001:16d8:ee71:2:0:5efe:192.168.1.210</ip>
                <descr>IPv6 Debian</descr>
        </hosts>
        <hosts>
                <host>isatap</host>
                <domain>example.com</domain>
                <ip>192.168.1.230</ip>
                <descr>ISATAP router</descr>
        </hosts>
        <hosts>
```

```
                <host>m0n0wall</host>
                <domain>example.com</domain>
                <ip>2001:16d8:ee71:1::1</ip>
                <descr>IPv6 m0n0wall</descr>
        </hosts>
        <hosts>
                <host>m0n0wall</host>
                <domain>example.com</domain>
                <ip>192.168.1.1</ip>
                <descr>IPv4 m0n0wall</descr>
        </hosts>
        <hosts>
                <host>win-server</host>
                <domain>example.com</domain>
                <ip>192.168.1.220</ip>
                <descr>IPv4 Windows</descr>
        </hosts>
        <hosts>
                <host>win-server</host>
                <domain>example.com</domain>
                <ip>2001:16d8:ee71:2:0:5efe:192.168.1.220</ip>
                <descr>IPv6 Windows</descr>
        </hosts>
</dnsmasq>
<snmpd>
        <syslocation/>
        <syscontact/>
        <rocommunity>public</rocommunity>
</snmpd>
<diag/>
<bridge/>
<syslog>
        <reverse/>
        <nentries>50</nentries>
        <remoteserver/>
        <remoteport/>
</syslog>
<nat>
        <rule>
                <protocol>tcp</protocol>
                <external-port>22</external-port>
                <target>192.168.1.210</target>
                <local-port>22</local-port>
                <interface>wan</interface>
                <descr/>
        </rule>
        <rule>
                <protocol>tcp</protocol>
                <external-port>80</external-port>
                <target>192.168.1.210</target>
                <local-port>80</local-port>
                <interface>wan</interface>
                <descr/>
```

```
        </rule>
        <rule>
                <protocol>tcp</protocol>
                <external-port>3389</external-port>
                <target>192.168.1.220</target>
                <local-port>3389</local-port>
                <interface>wan</interface>
                <descr>Remote Desktop</descr>
        </rule>
        <portrange-low/>
        <portrange-high/>
</nat>
<filter>
        <rule>
                <type>pass</type>
                <interface>wan</interface>
                <protocol>tcp</protocol>
                <source>
                        <any/>
                </source>
                <destination>
                        <address>192.168.1.210</address>
                        <port>22</port>
                </destination>
                <descr>NAT </descr>
        </rule>
        <rule>
                <type>pass</type>
                <interface>wan</interface>
                <protocol>tcp</protocol>
                <source>
                        <any/>
                </source>
                <destination>
                        <address>192.168.1.210</address>
                        <port>80</port>
                </destination>
                <descr>NAT </descr>
        </rule>
        <rule>
                <type>pass</type>
                <interface>wan</interface>
                <protocol>tcp</protocol>
                <source>
                        <any/>
                </source>
                <destination>
                        <address>192.168.1.220</address>
                        <port>3389</port>
                </destination>
                <descr>NAT Remote Desktop</descr>
        </rule>
        <rule>
```

```
                <type>pass</type>
                <descr>Default LAN -&gt; any</descr>
                <interface>lan</interface>
                <source>
                        <network>lan</network>
                </source>
                <destination>
                        <any/>
                </destination>
        </rule>
        <rule>
                <type>pass</type>
                <descr>Default IPsec VPN</descr>
                <interface>ipsec</interface>
                <source>
                        <any/>
                </source>
                <destination>
                        <any/>
                </destination>
        </rule>
        <tcpidletimeout/>
        <rule6>
                <type>pass</type>
                <interface>wan</interface>
                <protocol>tcp</protocol>
                <source>
                        <any/>
                </source>
                <destination>
                        <address>2001:16d8:ee71:2:0:5efe:192.168.1.210</address>
                        <port>22</port>
                </destination>
                <descr>SSH</descr>
        </rule6>
        <rule6>
                <type>pass</type>
                <interface>wan</interface>
                <protocol>tcp</protocol>
                <source>
                        <any/>
                </source>
                <destination>
                        <address>2001:16d8:ee71:2:0:5efe:192.168.1.210</address>
                        <port>80</port>
                </destination>
                <descr>HTTP</descr>
        </rule6>
        <rule6>
                <type>pass</type>
                <interface>lan</interface>
                <source>
                        <address>2001:16d8:ee71::/48</address>
```

```
                        </source>
                        <destination>
                                <any/>
                        </destination>
                        <descr/>
                </rule6>
                <rule6>
                        <type>pass</type>
                        <interface>wan</interface>
                        <protocol>tcp</protocol>
                        <source>
                                <any/>
                        </source>
                        <destination>
                                <address>2001:16d8:ee71:2:0:5efe:192.168.1.220</address>
                                <port>3389</port>
                        </destination>
                        <descr>Remote Desktop</descr>
                </rule6>
        </filter>
        <shaper/>
        <ipsec>
                <dns-interval/>
        </ipsec>
        <aliases/>
        <proxyarp/>
        <wol/>
</m0n0wall>
```

A.2 Linux configuration files

Unix-derived systems makes extensive use of configuration files when setting up services and properties. This section contains the ones described in this thesis.

A.2.1 Debian static configuration 6to4

/etc/network/interfaces

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# IPv4 static configuration eth0
allow-hotplug eth0
iface eth0 inet static
address 192.168.1.210
network 192.168.1.0
netmask 255.255.255.0
broadcast 192.168.0.255
gateway 192.168.1.1

#IPv6 static configuration eth0
iface eth0 inet6 static
```

```
pre-up modprobe ipv6
address 2002:8027:50aa:1::10
netmask 64
gateway 2002:8027:50aa:1::1
```

/etc/resolv.conf

```
domain example.com
search example.com
nameserver 2002:8027:50aa:1::1
nameserver 192.168.1.1
```

A.2.2 Ubuntu 10.04 ISATAP router

/etc/network/interfaces

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface   eth0 inet static
        address 192.168.1.230
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
        gateway 192.168.1.1

#IPv6 static configuration eth0
iface   eth0 inet6 static
        address 2001:16d8:ee71:1::10
        netmask 64
        gateway 2001:16d8:ee71:1::1
```

/etc/rc.local

```
#!/bin/sh -e
#
# rc.local

ip addr add 2001:16d8:ee71:2:0:5efe:192.168.1.230/64 dev is0
sysctl net.ipv6.conf.all.forwarding=1
/etc/init.d/radvd restart

exit 0
```

/etc/radvd.conf

```
interface is0
{
 AdvSendAdvert on;
 UnicastOnly on;
 AdvHomeAgentFlag off;
```

```
 prefix 2001:16d8:ee71:2::/64
 {
 AdvOnLink on;
 AdvAutonomous on;
 AdvRouterAddr off;
 };
};
```