

MASTER OF SCIENCE THESIS

# *A Transition Box*

By  
Karwan Sorani

*Stockholm, February 2001*

---

**Examiner and Adviser:**  
Björn Pehrson and Lars Albertsson  
Royal Institute of Technology (KTH)

**Supervisor:**  
Eva Jonsson  
Nortel Networks

*We would like to thank  
Eva Jonsson, for her advice, direction and support during the project.  
We would also like to thank  
Routing Architecture Lab for their help making this project successful.*

## *Abstract*

*It is expected that in the nearest future the usage of Internet Protocol version 6 (IPv6) in the Internet increase substantially. This is mainly due to the limitation in the IPv4 address space.*

*It is hard to see deploying IPv6 without some promise that old legacy networks will still be capable of connecting to the new IPv6 Internet, that because of the already existence of the massive amount of subnet for IPv4.*

*In this thesis we present a mechanism that have been proposed to provide transparency between IPv4 and IPv6 end-to-end solution. We also present a mechanism for address assignment, which makes this transition possible. We show how one could use such a mechanism to transparently establish hybrid communication and discuss some important issues like the complexities involved in building and deploying such a system.*

*This mechanism can cooperate with a number of applications such as Session Initiation Protocol (SIP). This cooperation is possible only in conjunction with an application level gateway (ALG). The project therefore includes an implementation proposal of a SIP- ALG.*

## Contents

<b>1 INTRODUCTION.....</b>	<b>6</b>
1.1 BACKGROUND .....	6
1.2 NORTEL NETWORKS INTEREST.....	8
1.3 SCOPE OF THE THESIS .....	8
1.4 STRUCTURE OF THE REPORT .....	8
<b>2 INTERNET PROTOCOL VERSION 6.....</b>	<b>9</b>
2.1 IPV6 OVERVIEW .....	9
2.2 IPV6 IMPROVEMENTS .....	9
<b>3 OVERVIEW OF TRANSITION METHODS.....</b>	<b>10</b>
3.1 THE DUAL STACK.....	12
<b>4 A TRANSITION BOX.....</b>	<b>14</b>
4.1 6TO4 .....	15
4.1.1 6to4 Address.....	17
4.1.2 Encapsulation of IPv6 packets.....	18
4.1.3 Simplest Use of 6to4.....	18
4.1.3.1 Sending and Receiving Rules for 6to4.....	19
4.1.3.2 The Return Path and Source Address Selection.....	20
4.1.4 More Complex 6to4 Usage Scenarios.....	20
4.1.4.1 The 6to4 Relay .....	21
4.1.5 Summarizing 6to4 .....	22
4.1.6 Other tunneling methods .....	23
4.1.6.1 6over4.....	23
4.1.6.2 Automatic tunnels .....	23
4.1.6.3 Tunnel Broker .....	23
4.2 INTRODUCTION TO NAT.....	24
4.2.1 Static NAT.....	25
4.2.2 Dynamic NAT.....	26
4.2.3 NAPT-PT.....	26
4.2.3.1 Address Translation .....	27
4.2.3.1.1 Address Binding.....	27
4.2.3.1.2 Address Lookup and Translation .....	28
4.2.3.1.3 Address Unbinding.....	28
4.2.3.2 Protocol Translation .....	28
4.2.3.2.1 IP Translation.....	28
4.2.3.2.2 ICMP Translation.....	30
4.2.3.2.3 Adjusting Checksum Values .....	31
4.2.3.3 ALG (Application Level Gateway).....	31
4.2.3.3.1 DNS-ALG .....	32
4.2.3.3.1.1 DNSSEC .....	32
4.2.3.3.2 FTP-ALG .....	33
4.2.3.3.3 SIP-ALG .....	34
4.2.3.3.3.1 SIP Overview .....	34
4.2.3 NAPT-PT Scenarios .....	35

---

4.2.4	<i>NAPT-PT limitations</i>	36
4.2.5	<i>Alternative methods to NAPT-PT</i>	37
4.2.5.1	RSIP	37
4.2.5.2	BIS	37
4.2.5.3	SOCKS64	38
4.2.6	<i>Summarizing NAPT-PT</i>	38
<b>5</b>	<b>EXPERIMENTAL SETUP</b>	<b>38</b>
5.1	LAB NETWORK	39
5.2	HARDWARE AND SOFTWARE USED	39
5.3	CONFIGURATION AND SETUP	39
5.5	FREEBSD	41
5.5.1	<i>KAME IPv6 stack</i>	41
5.5.2	<i>The Berkeley Packet Filter</i>	41
5.5.3	<i>Enabling 6to4 in FreeBSD</i>	42
<b>6</b>	<b>SIP-ALG IMPLEMENTATION</b>	<b>43</b>
<b>7</b>	<b>MODEL LIMITATION AND FUTURE WORK</b>	<b>44</b>
<b>8</b>	<b>CONCLUSIONS</b>	<b>45</b>
<b>9</b>	<b>ACRONYMS</b>	<b>46</b>
<b>10</b>	<b>TERMINOLOGY</b>	<b>48</b>
<b>11</b>	<b>REFERENCE</b>	<b>51</b>
<b>12</b>	<b>APPENDIX A</b>	<b>54</b>
<b>13</b>	<b>APPENDIX B</b>	<b>56</b>
<b>14</b>	<b>APPENDIX C</b>	<b>59</b>
<b>15</b>	<b>APPENDIX D</b>	<b>60</b>
<b>16</b>	<b>APPENDIX E</b>	<b>61</b>

# 1 Introduction

## 1.1 Background

Today the World Wide Web and the Internet are quickly becoming everywhere in our modern life. In the world at present, there are about six billion people and in many parts of the world the development of the computer technology has not come far. There is estimation that today there are about 350 million computers, 480 million mobile telephones, 600 million cars and 1 billion television sets. More and more people will be using wireless and portable computers, such as the Personal Digital Assistant (PDA<sup>1</sup>), and the use of the computers will decrease. With the huge increase of the number of PDA's, the number of mobile telephones and wireless PDA is expected to reach 1 billion by 2003 [70]. These have to be interconnected and connected to the Internet, and this development will lead to needing more IP addresses, which will enforce the expansion of Internet Protocol Version 6 (IPv6)<sup>2</sup>, due to the limitation in number of IPv4 addresses.

The Internet operates using the Internet protocol (IP) as the base. This protocol has worked well for number of years. All computers connected to the Internet must have an IPv4 address in order for them to be contacted. The number of possible combinations of IPv4 addresses is about four billion nodes, but in practice there are only two billion IPv4 addresses available [67]. However, IPv4 is showing its age as unexpected growth problems have appeared. Most notable is the rapid consumption of network address and the explosion in performance requirement for intermediate routing nodes. IPv4 will eventually be unable to support additional nodes or the requirements of new applications. This because of the fact that IPv4 was introduced when exceeding the theoretical maximum number of four billion nodes was not a possibility. With the advancements and the explosion of computer technology that limit will soon be reached [18].

### Why IPv6?

The number of IPv6 nodes is growing and the number of IPv4 nodes grows as well until it runs out of addresses. This leads to the situation where you have an IPv4 section of the Internet, which is very large but losing in overall market share. As a result of the huge increase of IPv6 deployment, there will be an equally large increase in the number of routers and infrastructure devices needed. All of these devices will need to be IP addressable. The larger IPv6 address space will be necessary to support this kind of growth, since IPv6 offers a larger number of devices to be uniquely addressed and individually locatable in the world. No longer will computers have to be turned off so that IP addresses can be shared [58].

Third Generation Partnership Project 3GPP is a standardization forum for Third Generation (3G) mobile systems with the aim to make IPv6 as the protocol for

---

<sup>1</sup> A list of Acronyms and Terminology can be found at the end of the report.

<sup>2</sup> Known first as IPng and then as IPv6.

future IP multimedia services. 3GPP will develop the specifications for next-generation mobile communication. 3G mobile communications will offer user benefits such as advanced mobile Internet and multimedia services that require wideband capabilities and high-speed data transfer [68]. The 3GPP says that you must have IPv6 addresses in your 3GPP user equipment. A lot of telephones will involve voice and data, and if these two should work together, the use of IPv6 should become the only reasonable alternative [59].

IPv6 is a new network protocol that features improved scalability and routing, security, ease-of-configuration and higher performance compared to IPv4. In addition, IPv6 has important mobility and security features that IPv4 can't support. However IPv4 and IPv6 packets can coexist on a link with the help of a transition mechanism.

In many countries the increase to Internet connectivity is huge. The amount of IPv4 addresses available for these markets is insufficient to meet the demand. China, for example has far less address space than Nortel Networks. This demand points to a need for the increased address space of IPv6, which led to the development of a next-generation Internet Protocol [1], known first as IPng and then as IPv6, by the IETF for several years to replace the current Internet Protocol known as IPv4 and overcome limitations, mentioned above, imposed by IPv4.

Among other improvements they considerably increased the address space so that it will hopefully last for the next few decades. Since developing a new protocol, which in addition is of such major importance as the IP protocol takes some years and the migration can also not be done in a day it was clear that we would still have to live with IPv4 for a couple of years. That meant to find solutions to the problem of limited address space, since this was the most pressing problem.

Moreover, the use of new features, such as anycast addresses, will improve network utilization, increasing the productivity of all activities that make use of the Internet, while the streamlined IPv6 packet header will have a similar effect by increasing the performance of router software. All these factors taken together make a strong case for IPv6.

IPv4, however, is not about to go away "over night" as the IPv6 systems are rolled in. As a consequence, it is necessary to develop transition mechanisms that enable applications to continue working while the hosts and networks are being upgraded.

So the conclusion is that due to the expansion of wireless standards using IPv6 and the maintaining compatibility with IPv4, there have to be a transition mechanism to ensure a smooth transition between IPv4 and IPv6. This paper describes a solution to such a transition problem.

## *1.2 Nortel Networks interest*

A clear gap in Nortel's product portfolio is a Transition Box (TB), which can handle the transition between IPv6 and IPv4. This Transition Box ensures a transparent communication environment baring the two protocols.

The Transition Box will allow an enterprise or other island of IPv6 functionality to access other islands of IPv6 across the sea of the existing IPv4 world as well as access legacy servers in the IPv4 domain [45].

The conclusion is that Nortel has to make their product portfolio IPv6 capable, concentrating on the products that are likely to be deployed in the 3G models. Without an IPv6 solution, Nortel Networks will not be able to compete in the 3G wireless markets.

## *1.3 Scope of the thesis*

This Master Thesis studies a transition mechanism with focus on the protocol translation IPv6/IPv4 which will provide the interconnect functionality between IPv6 and IPv4 domains. It is also includes a solution for such a transition mechanism and discusses alternative approaches.

The practical part of the Master Thesis includes an implementation of Session Initiation Protocol-Application Level Gateway (SIP-ALG) based on FreeBSD Operative System.

## *1.4 Structure of the Report*

Chapter 1 gives a brief overview of the needs of IPv6 in the future and background of the Transition Box project in Nortel Networks.

Chapter 2 gives a brief introduction to Internet protocols and IPv6's improvements.

Chapter 3 gives a brief overview of a number of well-known transition mechanisms.

Chapter 4 introduces and describes the Transition Box project and the technology used in this project, such as tunneling and network address and protocol translation.

Chapter 5 describes the equipments we used to set up the experimental environment, and describe the prototype implementation of the model introduced in chapter 4.

Chapter 6 gives a brief evaluation of our SIP implementation that is a result of this project.

Chapter 7 describes the limitation and future work in this area.

Chapter 8 concludes the work and gives suggestions on the future development.

Chapter 9 introduces a list of acronyms used in this report.

Chapter 10 introduces and describes a list of terms used in this report.

## **2 Internet protocol version 6**

### *2.1 IPv6 Overview*

Since 1992 the new protocol IPv6, has been in development with the intent of replacing today's IPv4. It was when network experts realized that the nature of the Internet had changed and was in the need of major improvements. IPv4 has its roots in the early 1970s when the Internet consisted only of limited research networks.

This new protocol brings new functionality into internetworking and better performance through optimized headers (see next section). When designing IPv6, much research was made to assure that IPv6 would handle the expected growth of the Internet and all the new services that will follow with it.

In the next section, IPv6 features will be briefly described. For further information, please refer to [1].

### *2.2 IPv6 Improvements*

IPv6 was designed to be an upgrade from IPv4, and with IPv6 a lot of improvements were made. Functions that were not used very much or did not work very well with IPv4 were removed in IPv6. Functions that worked well were kept and new features that were needed were added to IPv6. IPv6 has been improved compared to IPv4 and some improvements are described below.

#### *Expanded Addressing and Routing*

The IP address size is increased from 32 bits to 128 bits, which will support more levels of addressing hierarchy, a much greater number of addressable nodes and simpler auto-configuration of addresses. The hierarchical addresses leads to smaller routing tables, less routing traffic and faster route lookups. A new type of address called anycast address is defined, used to send a packet to any one of a group of nodes.

#### *Improved support for extensions and options*

The changes in the IP header options allows for more efficient forwarding and greater flexibility for introducing new options in the future.

#### *Authentication and Security*

Extension to support authentication, data integrity and data confidentiality are specified for IPv6. IPv6 can authenticate the complete IP header, including options [57].

*Improved support for auto-configuration*

IPv6 support many forms of auto-configuration, from plug and play configuration of node addresses on an isolated network to the services offered by the Dynamic Host Configuration Protocol (DHCP).

*Simplified header format*

The changes in the header format from IPv4 to IPv6 makes the packet processing more efficient and reduces the common-case processing cost of packet handling.

*Minimized Administration*

In IPv6 the network devices are allowed to configure themselves automatically, which decreases the probability of configuration errors.

For more detailed information about IPv6 improvements, see [1].

### **3 Overview of Transition Methods**

Over the coming years there will probably be a gradual transition of the Internet from IPv4 to IPv6. It is unreasonable to expect the millions of IPv4 nodes to covert, overnight, to IPv6. Thus, during this migration period it is important that existing IPv4 application continue to work with newer IPv6 applications. For example, a vendor cannot provide a Telnet client that works only with IPv6 Telnet servers but must provide one that works with IPv4 servers and one that work with IPv6 servers. Better yet would be one IPv6 Telnet client that work with both IPv4 and IPv6 servers [4]. In this chapter we will present a number of transition methods.

The transition mechanisms can be categorized depending on the particular transition scenario being studied. For communication between IPv6-only islands, using the tunneling mechanisms is the suitable choice, since it is easy to implement and no translation on network layer is required. To accomplish transparency communication between IPv4 sites and IPv6 sites other mechanism, such as BIS and NAT-PT, are used see Figure 3.1. This figure, which is based on a Internet Engineering Task Force (IETF) draft [33], shows some of the most common mechanism that is in use now.

The current transition efforts are discussed at the *IETF IPng Transition Working Group* (ngtrans). IPng's migration strategy relies on three major components namely Dual Stack, Tunneling and Header translation [2].

Obviously, systems must have the ability to understand IPv6, with which they can use tunneling to reach other IPv6 systems, even across an IPv4 network. Header translation offers a way for those systems to maintain contact with older IPv4 hosts.

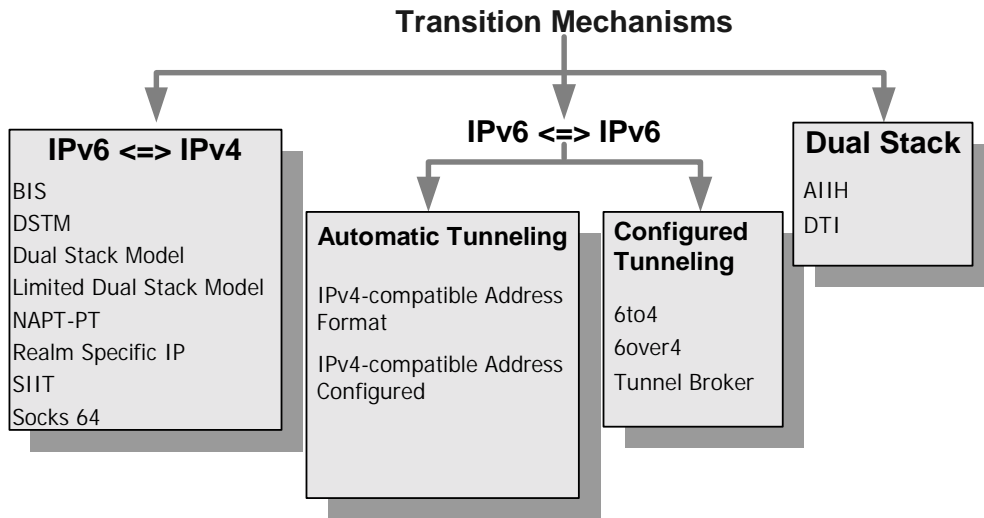
The first step towards the IPv6 is the deployment of systems that support IPv6. Those systems are not likely to have many other systems with which to communicate, most other systems will still be using IPv4. These new IPv6

systems, therefore, will likely be dual stack systems, which are able to use both IPv6 and IPv4. Dual stack systems can use IPv6 to communicate with IPv6 system, and they can also fall back to IPv4 to communicate with older systems. Later in this chapter we present the dual stack mechanism.

There will be more systems that want to use IPv6 to communicate. Unfortunately, older IPv4 networks may separate those IPv6 systems, and to reach each other, the IPv6 systems have to tunnel through the IPv4 network. To support the transition, the standards specify IPv4-compatible addresses and IPv6-over IPv4 tunneling. An IPv4 compatible address is identified by having a 96-bit zero prefix followed by the 32-bit IPv4 address of the node. Both of these mechanisms allow IPv6 to operate over IPv4 networks. The most straightforward approach to tunneling relies on IPv4-compatible addresses. When an IPv6 datagram reaches the boundary of the IPv4 network, the router encapsulates it in an IPv4 datagram. As an IPv4 datagram, the new message must have an IPv4 destination address. To derive that address, the router extracts the IPv4 address embedded in the IPv6 address. For example, to send an IPv6 packet to the IPv4-compatible address `::128.173.91.72`, the source node would encapsulate the IPv6 packet in an IPv4 packet and send the IPv4 packet to the IPv4 host `128.173.91.72`.

Automatic tunneling is another transition mechanism. This is used to connect IPv6 sites or hosts separated by an IPv4 infrastructure. Automatic tunnels can only be created between IPv6 hosts or sites with IPv4-compatible IPv6 addresses. When IPv4-compatible addresses are unavailable, automatic tunneling is not possible. In these situations, configured tunneling can be used instead. Some configured tunneling mechanism requires explicit configuration at the entry point to the IPv4 network. That configuration must specify the IPv4 destination to be used for the tunneled packet. Since the IPv6 destination is not an IPv4-compatible address, routers cannot automatically derive an IPv4 destination [2].

Another potential transition mechanism is to translate IPv4 headers to IPv6 headers and vice versa [46]. This gives the appearance that the IPv4 node is communicating with an IPv6 node.



*Figure 3.1: Transition Mechanisms*

These mechanisms are planned as a transition tool used during the period of co-existence of IPv4 and IPv6. The most important gain from these mechanisms is the possibility for new IPv6 sites for example IPv6-only sites to communicate with existing sites such as IPv4 backbone and to communicate to other IPv6-only sites over IPv4 sites. In general those mechanisms make it possible to establish communication between mixture sites.

The transition mechanisms that we chose to study in our project are:

- 6to4, which handles the connectivity between two IPv6 island only. Unlike 6over4 it is not required an IPv4 multicast infrastructure and there is no need of IPv4-compatible addresses like automatic tunneling.
- NAPT-PT, where IPv6 nodes are allowed to communicate with the IPv4 nodes transparently using a single IPv4 address. This is unlike Stateless IP/ICMP Translator (SIIT) that requires as many IPv4 addresses as the number of IPv6 nodes.

Before we discuss these two transition mechanisms that will be the main part of this report, we present one of the most common transition mechanism in use in the following section.

### **3.1 The dual stack**

Dual stack is a technique for providing complete support for both Internet protocols -- IPv4 and IPv6 -- in hosts and routers. It can be used as a first step in migration to IPv6 by deployment of systems that support IPv6. At least at first, those systems are not likely to have many other systems with which to communicate. Most other systems will still be using IPv4. These new systems, therefore, will still likely be dual stack systems, able to use both IPv6 and IPv4.

Figure 3.1 shows an example of an IPv6 server on dual stack host serving IPv4 and IPv6 clients. This dual stack provides a complete IPv6 and IPv4 implementation called “IPv6/IPv4 nodes”. IPv6/IPv4 nodes have the ability to send and receive both IPv4 and IPv6 packets. They can directly interoperate with IPv4 and IPv6 nodes using IPv4 and IPv6 packets respectively.

Even if a node may be set to support both protocols, one or the other stack may be disabled. Thus IPv6/IPv4 nodes may be operated in one of three nodes:

- With their IPv4 stack enabled and their IPv6 stack disabled.
- With their IPv6 stack enabled and their IPv4 stack disabled.
- With both stacks enabled.

IPv6/IPv4 nodes with their IPv6 stack disabled will operate like IPv4-only nodes. Similarly, IPv6/IPv4 nodes with their IPv4 stacks disabled will operate like IPv6-only nodes [20].

In the following example we have an IPv6 server on dual stack host serving IPv4 and IPv6 clients. Such hosts will probably run like this for many years during the transition to IPv6. At some point many systems will be able to turn off their IPv4 stack, but only time will tell when that will occur.

We assume the clients and server are on the same Ethernet. Routers could also connect them, as long as all the routers support IPv4 and IPv6. Both clients can establish a connection with the server by sending an SYN segments. The IPv4 client host will send the SYN in an IPv4 datagram and the IPv6 client host will send SYN in an IPv6 datagram. The TCP segment from the IPv4 client appears on the wire as an Ethernet header followed by an IPv4 header, a TCP header, and the TCP data. The Ethernet header contains a type field of 0x0800, which identifies the frame as an IPv4 frame. The TCP header contains the destination port of 8888.

The destination IP address in the IPv4 header, not shown, would be 206.62.226.42.

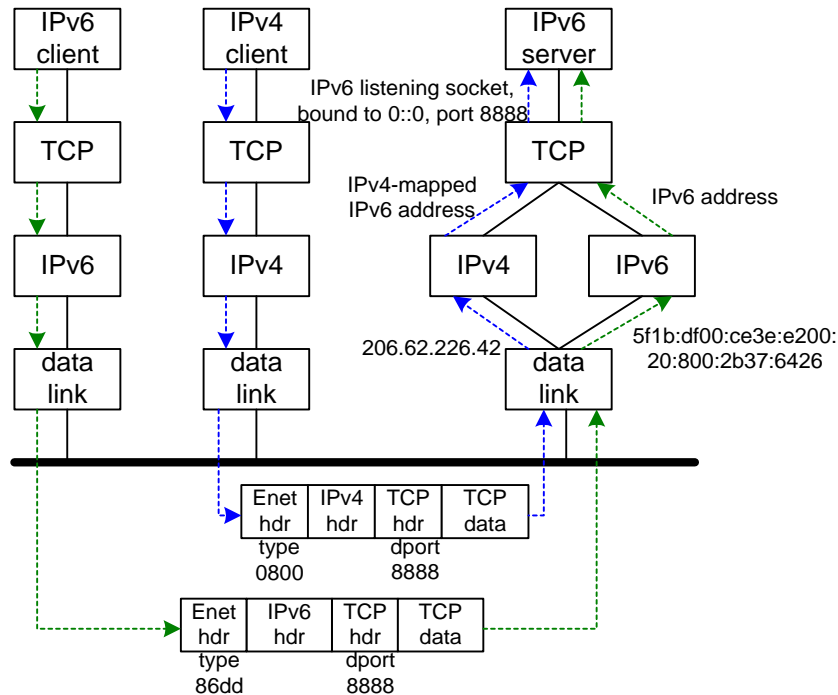


Figure 3.2: IPv6 server on dual-stack host serving IPv4 and IPv6 clients.

The TCP segment from the IPv6 client appears on the wire as an Ethernet header followed by an IPv6 header, a TCP header, and the TCP data. The Ethernet header contains a type field of 0x86dd, which identifies the frame as an IPv6 frame. The TCP header has the same format as the TCP header in the IPv4 packet and contains the destination port of 8888. The destination IP address in the IPv6 header, which we do not show, would be 5f1b:df00:ce3e:e200:20:800:2b37:6426 [4].

The main drawbacks of this method are the following. An IPv4 address must be allocated to each equipment. If we map this onto our transition model, it means that the IPv6 domain is included in the IPv4 world. Routers must be configured for the two protocols and IPv4 applications must be slightly modified and recompiled to be adapted to the IPv6 API.

## 4 A Transition Box

The IPv6 global Internet as of today is mostly built using tunnels over the existing IPv4 infrastructure. Those tunnels are difficult to configure and maintain in a large-scale environment. The 6bone has proven that large sites and ISPs can do it, but this process is too complex for the isolated end user who already has an IPv4 connection and would like to enter the IPv6 world. Thus, introducing Transition Box (see Figure 4.1) is a good idea to solve the connectivity problem between IPv6 island only and IPv4 island only and the connectivity problem between two IPv6 island only via an IPv4 island only.

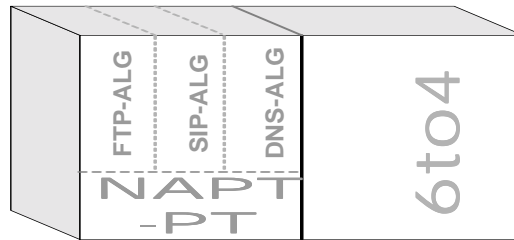


Figure 4.1: A Transition Box

This Transition Box is intended as a start-up transition tool used during the period of migration from IPv4 to IPv6. Figure 4.2 illustrates the some places in the network where TB can be used.

The Transition Box consists of two main mechanisms. A “6to4” mechanism that handles the connectivity between two IPv6 island only, and a “NAPT-PT” mechanism that handles the connectivity between IPv6 island only and IPv4 island only. The following section we will discuss these two mechanisms.

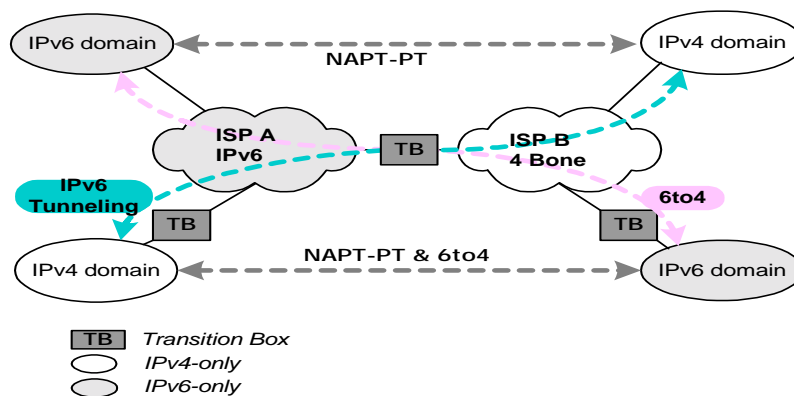


Figure 4.2: Utilizing the Transition Box

### 4.1 6to4

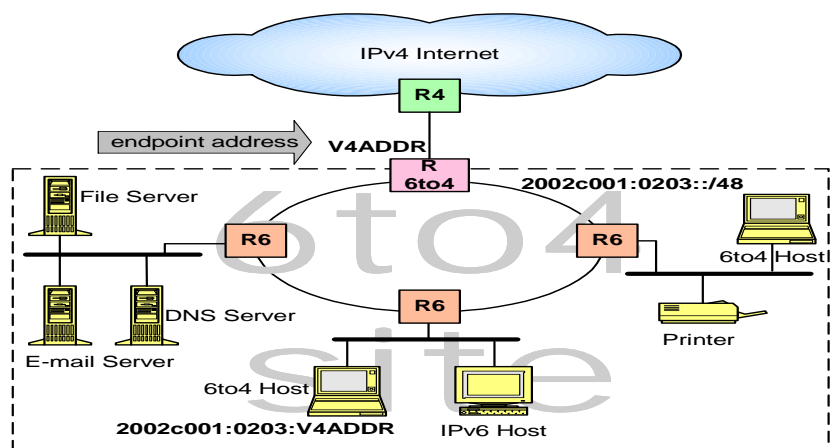
6to4 is a method for connecting IPv6 sites over the existing IPv4 Internet infrastructure. It is used for assigning a temporary unique IPv6 address prefix to any site that currently has at least one globally unique IPv4 address, and specifies an encapsulation mechanism for transmitting IPv6 packets using such a prefix over the global IPv4 network. It is important to keep in mind that this mechanism is not intended as a permanent solution. It is a start-up transition tool used during the period of co-existence of IPv4 and IPv6 [32].

Although the mechanism is specified for an IPv6 site, it can equally be applied to an individual IPv6 host or very small site, as long as it has at least one globally unique IPv4 address. 6to4 can be used to communicate directly with other 6to4

sites. It can also be used as a 6to4 relay to communicate with 6bone sites. So with this mechanism it is possible for isolated IPv6 sites, attached to an IPv4 network that has no native IPv6 support, to communicate with other such IPv6 sites with minimal manual configuration, before they obtain native IPv6 connectivity. This mechanism provides a solution to the complexity problem of using manually configured tunnels by specifying a unique routing prefix for each end-user site that carries an IPv4 tunnel endpoint address.

IPv6 sites or hosts connected using this method does not require IPv4-compatible IPv6 addresses [38] or configured tunnels. In this way IPv6 gains considerable independence of the underlying wide area network and can step over many hops of IPv4 subnets. On the other hand 6to4 method does not meet the requirements set for the service require IPv6 connectivity with globally routable address [7], since 6to4 does not use this address. 6to4 requires a border router on the site for the minimal configuration feature to be valid.

The shortened name of this mechanism is 6to4 (not to be confused with 6OVER4). Figure 4.3 illustrates an example of a typical 6to4 site.



*Figure 4.3: A 6to4 site*

The site's IPv4 tunnel endpoint will be advertised (to be used for a dynamic tunnel) in a special external routing prefix (will be discussed in later section) for that site. Thus one site trying to reach another will discover the 6to4 tunnel endpoint from a *Domain Name System* (DNS) name to address lookup and use a dynamically built tunnel from site to site for the communication. The tunnels are transient in that there is no state maintained for them, lasting only as long as a specific transaction uses the path. A 6to4 tunnel also bypasses the need to establish a tunnel to a wide-area IPv6 routing infrastructure, such as the 6bone [51].

### 4.1.1 6to4 Address

A 6to4 address is a special type of IPv6 unicast addresses for addressing of *6to4 hosts*. The prefix of an 6to4 address has exactly the same format as normal 48-bit prefixes in the IPv6 *Aggregatable Global Unicast Address* [7] and such prefix provides enough space to hold the 32 bits required for the 32-bit IPv4 tunnel endpoint address (called V4ADDR). This prefix can be abbreviated as 2002:V4ADDR::/48 and it stands for the Public Topology [32] (see Figure 4.4).

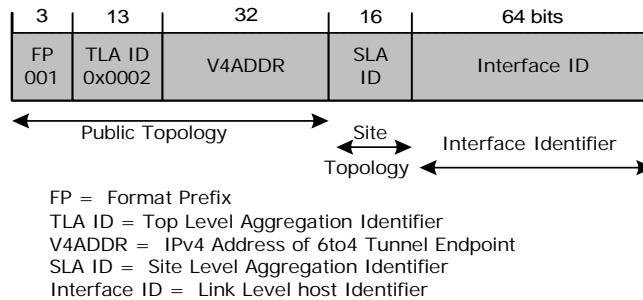


Figure 4.4: 6to4 Prefix Format

The IANA has permanently assigned one 13-bit IPv6 Top Level Aggregator (TLA) identifier under the IPv6 Format Prefix 001 [7] for the 6to4 scheme. Its numeric value is 0x0002, i.e., it is 2002::/16 when expressed as an IPv6 address prefix, this is illustrated in Figure 4.5.

Binary Value	Hex Value	IPv6 Prefix	Assignment
0 0000 0000 0010	0x0002	2002::/16	6to4

Figure 4.5: TLA ID format for 6to4 address

Address selection is an important issue when 6to4 addresses are used. In order to ensure the correct operation of 6to4 in complex topologies, source and destination address selection must be appropriately implemented. If the source IPv6 host sending a packet has at least one 2002:: address assigned to it, and if the set of IPv6 addresses returned by the DNS for the destination host contains at least one 2002:: address, then the source host must make an appropriate choice of the source and destination addresses to be used. The mechanisms for address selection in general are under study [37]. Subject to those general mechanisms, the principle that will normally allow correct operation of 6to4 is as follows:

- One host has only a 6to4 address, and the other one has both a 6to4 and a native IPv6 address, then the 6to4 address should be used for both.
- Both hosts have a 6to4 address and a native IPv6 address, then either the 6to4 address should be used for both, or the native IPv6 address should

be used for both. The choice should be configurable. The default configuration should be native IPv6 for both [32].

### 4.1.2 Encapsulation of IPv6 packets

When a host sends packets from its 6to4 site through the IPv4 interface, these packets leave the 6to4 site with the IPv6 packets encapsulated in IPv4 packets, see Figure 4.6. This IPv4 interface which carrying the 6to4 traffic is notionally equivalent to an IPv6 interface, and is referred to below as a pseudo-interface, although this phrase is not intended to define an implementation technique. The configuration of V4ADDR must complete on the IPv4 interface.

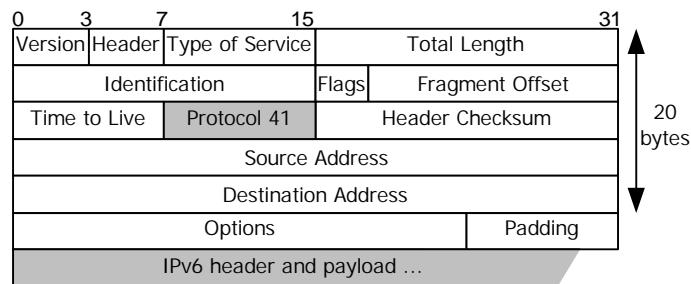


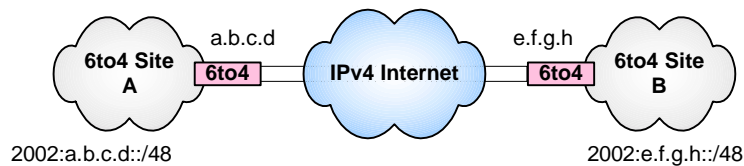
Figure 4.6: Encapsulation of IPv6 packet

The IPv4 protocol type field for encapsulated packets sets to 41. One or both of Source and Destinations addresses will be identical to the V4ADDR field of an IPv6 prefix formed as specified above (see section 4.1.1 for details). The IPv4 packet body contains the IPv6 header and payload. Furthermore the Time to Live (TTL) will be set as normal [23], as will the encapsulated IPv6 hop limit [24]. Other considerations are as described in Section 4.1.2 of [38].

### 4.1.3 Simplest Use of 6to4

The simplest scenario for 6to4 (Figure 4.7) is when several 6to4 sites are involved connected to the same IPv4 network, each of which has no less than one connection to a shared IPv4 Internet that can be the global Internet. In this case, there is no requirement that the sites all connect to the same Internet. The only requirement is that any of the sites is able to send IPv4 packets with protocol type 41 to any of the others.

By definition, each site has an IPv6 prefix in the format defined in Section 4.1.1. A new 6to4 site advertises the 6to4 prefix to its site via the Neighbor Discovery (ND) protocol [8], which will cause IPv6 hosts at this site to have their DNS name/address entries to include the 6to4 prefix for the site in them. For example, site A that owns IPv4 address a.b.c.d will create DNS records with the prefix (FP=001, TLA= 0x0002, NLA= a.b.c.d)/48 (i.e. 2002:a.b.c.d::/48). Site B that owns address e.f.g.h will create DNA records with the IPv6 prefix (FP=001, TLA= 0x0002, NLA=e.f.g.h)/48 (i.e. 2002:e.f.g.h::/48).



*Figure 4.7 Simple 6to4 scenario*

In operation, when one IPv6-enabled host at 6to4 site A tries to access an IPv6-enabled host by domain name at 6to4 site B, the DNS will return an IPv6 with prefix (FP=001, TLA= 0x0002, NLA= e.f.g.h)/48 and whatever SLA and Interface ID applies. The requesting host selects the IPv6 address and sends a packet off to its nearest router, eventually reaching its site boundary router (6to4 router).

When an outgoing packet reaches the 6to4 router, it is encapsulated as defined in Section 4.1.2, according to the additional sending rule defined in Section 4.1.3.1.1

#### **4.1.3.1 Sending and Receiving Rules for 6to4**

When the requesting site's 6to4 router sees that it must send a packet to another site (that is, there is a non-local destination), and that the next hop destination prefix contains the special 6to4 *Top Level Aggregation* (TLA) value of 2002::/16, the IPv6 packet is encapsulated in an IPv4 packet using an IPv4 protocol type of 41, as defined in the *Transition Mechanisms* [19]. The source IPv4 address will be the one in the requesting site's 6to4 prefix (which is the IPv4 address of the outgoing interface to the Internet on the 6to4 router, and contained in the source 6to4 prefix of the IPv6 packet), and the destination IPv4 address will be the one in the next hop destination 6to4 prefix of the IPv6 packet.

Within a 6to4 site, addresses with the 2002::/16 prefix, apart from those with the local 2002:V4ADDR::/48 prefix, will be handled like any other non-local IPv6 address, i.e., by a default or explicit route towards the 6to4 border router.

When the destination site's 6to4 router receives the IPv4 packet, and recognizes that it has an IPv4 protocol type of 41, IPv4 security checks are completed and the IPv4 header is removed, leaving the original IPv6 packet for local forwarding.

The sending rule above is the only modification to IPv6 forwarding, because the receiving rule was already specified for the basic IPv6 Transition Mechanism mentioned earlier [19]. Along with advertisement of the 6to4 prefix by appropriate entries in the DNS, any number of sites can interoperate without manual tunnel configuration.

It is not necessary to operate an exterior routing protocol (for instance, BGP4+) for 6to4 simple scenarios because the IPv4 exterior routing protocol is handling

this function. Also, no new entries in IPv4 routing tables result from the use of 6to4.

#### **4.1.3.2 The Return Path and Source Address Selection**

In order to have packet flow in both direction when two 6to4 sites communicate, it is essential that IPv6 packets sent use a packet with a 6to4 prefix as a source address when talking to a site with a 6to4 prefix; in other words, the destination must have a 6to4 prefix. This is not an issue in the simple scenario given above because both sites have only IPv4 connectivity, so they have 6to4 prefixes for their site to communicate with. DNS lookups for host systems at these sites will return an IPv6 address with a 6to4 prefix. Thus, source address selection is not an issue.

As we will soon see, source address selection is an issue for more complex 6to4 usage scenarios; therefore, some source address selection algorithm is necessary in IPv6 hosts. The exact form and method of the algorithm to use is under active study at the IETF IPv6 (ipng) working group [64]. Mean-while, for the purposes of understanding 6to4, it is sufficient to realize that when a 6to4 connected sending site is sending to a destination site using that site's 6to4 prefix, the sending host must guarantee that the source IPv6 address uses the sending site's 6to4 prefix.

#### **4.1.4 More Complex 6to4 Usage Scenarios**

During the transition to IPv6 it is expected to exist more complex scenarios than described above. In this section we describe some interesting 6to4 usage scenarios.

A site with only 6to4 connectivity tries to reach a site with both 6to4 and native IPv6 connectivity (see Figure 4.8); in this case the 6to4 address should be used for both [32]. Some host rules for choosing multiple destination addresses must result in the 6to4 address being chosen, because only a local 6to4 IPv6 source address is available. Of course source selection is not an issue in this case because there is only the 6to4 IPv6 address to use. In the same way, when a site with both 6to4 connectivity and native IPv6 connectivity tries to reach another site with only 6to4 connectivity, in which case the source address selection algorithm mentioned in Section 4.1.3.1.2 is essential to ensure that the site's 6to4 IPv6 address is chosen. No destination selection is required because there is only one choice, that is, 6to4.

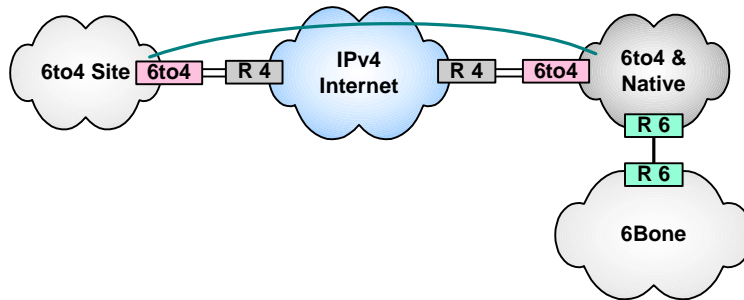


Figure 4.8: Connectivity between 6to4 site and 6to4-native site

Another variation of these scenarios is when a site with 6to4 and native IPv6 connectivity is trying to reach another site with only native IPv6 connectivity (see Figure 4.9), making a source address selection algorithm essential to make sure the site's native IPv6 address is chosen. No destination selection is required, because there is only one choice, that is, the native IPv6 address.

Similarly, when a site that has only native IPv6 connectivity tries to reach a site with 6to4 and native IPv6 connectivity, a host rule is essential for choosing among multiple addresses to ensure that a native IPv6 address is chosen, because only a local native IPv6 source address is available. Again, source selection is not an issue in this case because only the native IPv6 address can be used.

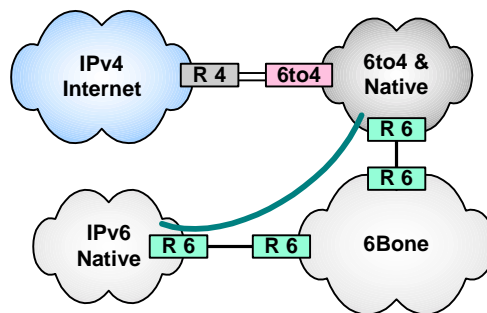


Figure 4.9: Connectivity between native site and 6to4-native site

If both sites have 6to4 and native IPv6 connectivity, then either the 6to4 address should be used for both, or the native IPv6 address should be used for both. The choice should be configurable. The default configuration should be native IPv6 for both [32].

#### 4.1.4.1 The 6to4 Relay

The most interesting, and most complex, 6to4 scenario is that of sites with only 6to4 connectivity communicating with sites with only native IPv6 connectivity. This is accomplished by the use of a 6to4 relay that supports both 6to4 and native IPv6 connectivity (see Figure 4.10). The 6to4 relay is simply a normal

router that happens to have at least one logical *6to4 pseudo-interface* and at least one other IPv6 interface. It is nothing more than an IPv4/IPv6 dual-stack router.

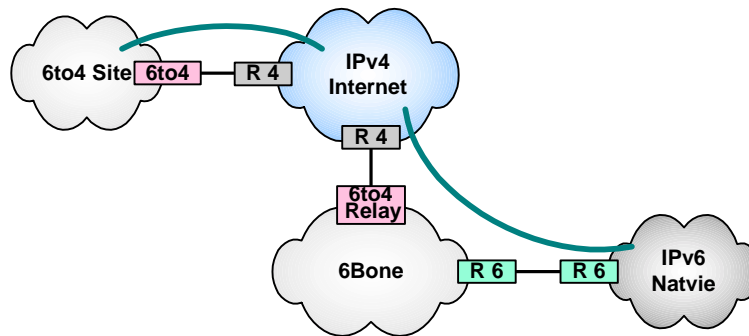


Figure 4.10: Connectivity between 6to4 site and native site

The 6to4 relay advertises a route to 2002::/16 for itself into the native IPv6 infrastructure it is attached to. The native IPv6 network operators must filter out and discard any 6to4 (2002:...) prefix advertisements longer than /16. In addition, the 6to4 relay may advertise into its 6to4 connection whatever native IPv6 routes its policies allow, which the 6to4 router at the 6to4-only site picks up with either a BGP4+ peering session, or with a default route, to the 6to4 relay.

Thus the 6to4-only site will try to send a packet to the native IPv6-only site by forwarding an encapsulated (tunneled) IPv6 packet to the 6to4 relay, which removes the IPv4 header (decapsulates) and forwards the packet on to the IPv6-only site.

Potentially, multiple 6to4 relays are needed, one for each separate IPv6 routing realm (collection of IPv6 routing ISPs). In practice, it is expected that all native IPv6 ISP services will be interconnected even if the use of inter-IPv6-ISP manually configured tunnels are required to do so. This is currently the case as of early 2000, because all 6bone 3FFE::/16 TLA networks and all production 2001::/16 sub TLA networks are interconnected with each other.

It is expected that native IPv6 service providers will choose to operate 6to4 relays as a simple extension of their service. There are no special rules or exceptions to 6to4 as described here for this to happen because the 6to4 relay is simply operated as part of an end-user site that belongs to the IPv6 ISP.

#### 4.1.5 Summarizing 6to4

6to4 is a powerful IPv6 transition tool that allow both traditional IPv4-based Internet end-user sites and new IPv6-only Internet sites to utilize IPv6 and operate successfully over the existing IPv4-based Internet routing infrastructure. The 6to4 mechanism allows isolated IPv6 routing domains to communicate with other IPv6 routing domains, even in the total absence of native IPv6 service providers.

**Advantages:**

- Does not break the end-to-end connectivity concept.
- Does not require the use of IPv4 compatible addresses or configured tunneling for encapsulation.

**Disadvantages:**

- Encapsulation adds an additional load to the network.
- Add complexity due to mixture of IPv6 and IPv4 addresses in the routing tables.

## **4.1.6 Other tunneling methods**

### **4.1.6.1 6over4**

6over4 interconnects isolated IPv6 hosts without the use of explicit tunnels [21]. The motivation behind this method is to allow isolated IPv6 hosts, which have no directly connected IPv6 router, to communicate. There is no need to IPv4-compatible addresses, and to determine the tunnel endpoint neighbor discovery protocol is used.

This method can work when there is an IPv4 multicast infrastructure. It requires at least one router to support IPv6 and IPv4 multicast to be operational within the domain. This restricts the possibility of utilizing this method for IPv6 over IPv4 connectivity from any point of attachment to the IPv4 Internet, as IPv4 multicast is not widely available.

### **4.1.6.2 Automatic tunnels**

Automatic tunnels connect IPv6 sites or hosts separated by an IPv4 infrastructure [19]. Automatic tunnels can only be created between IPv6 hosts or sites with IPv4 compatible IPv6 addresses. Whereas the tunnel endpoints have to be manually configured with configured tunnels, the tunnel endpoints are automatically derived from the IPv4 compatible IPv6 addresses. The two main caveats of automatic tunneling are that global IPv6 connectivity is not achieved; only two hosts with IPv4 compatible IPv6 addresses may communicate.

The problem with this method is that it does not solve the address exhaustion problem of IPv4. Also there is a great fear to include the complete IPv4 routing table into the IPv6 world because this would worsen the routing table size problem [39].

### **4.1.6.3 Tunnel Broker**

The Tunnel Broker is the place where the user connects to register and activate tunnels. The Tunnel Broker manages tunnel creation, modification and deletion on behalf of the user. It also can be seen as virtual IPv6 ISPs, providing IPv6 connectivity to users already connected to the IPv4 Internet. In the global IPv6 Internet it is expected that many tunnel brokers will be available so that the user will just have to pick one. The list of the tunnel brokers should be referenced on

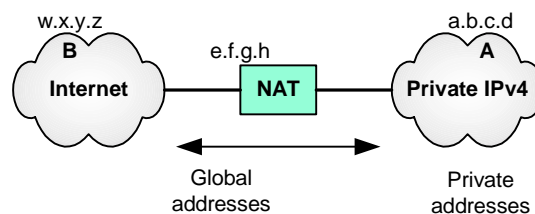
a "well known" web page (e.g. on <http://www.ipv6.org>) to allow users to choose the "closest" one, the "cheapest" one.

The motivation for the development of the tunnel broker model is to help early IPv6 adopters to hook up to an existing IPv6 network (e.g. the 6bone) and to get stable, permanent IPv6 addresses and DNS names. The concept of the tunnel broker was first presented at Orlando's IETF in December 1998 [39].

## 4.2 Introduction to NAT

The NAT protocol simplifies communication between the private network and the Internet. NAT is a device that translates IP addresses, converting non-unique IPv4 addresses to globally unique IPv4 addresses [16]. Usually the NAT device is used when site local addresses, that are not globally unique, are used internally within an organization. The NAT contains a pool of globally unique IPv4 addresses. When a packet enters the NAT the private organization address is replaced with a globally unique address from the pool.

A scenario is, as shown in figure 4.11, when host A, who is using a private address communicates with host B who is assigned a global address by its Internet Service Provider (ISP). As packets move across the NAT, the IP addresses in the IP header must be exchanged from a private address to a global address. After the exchange it will look to host B as if host A really has a global address and it will thus send any replies to that address.



*Figure 4.11: Network Address Translation*

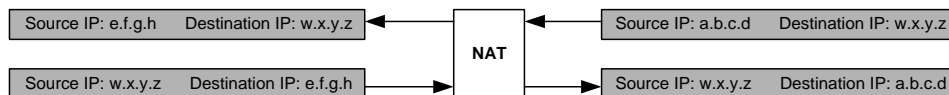
When the NAT changes the IP address in the IP header, the hosts will not notice the change, since NAT does the translation transparently. To be able to correctly map packets to the right host on the other side, an entry has to be set up in a translation table (Figure 4.12) by NAT, for every session between internal and external hosts.

Internal IP	External IP
a.b.c.d	e.f.g.h
.	.
.	.
.	.

*Figure 4.12: NAT-Table*

The internal IP address is the private IP address of the internal host and the external IP address is a public IP address on the NAT, which is assigned to map the private address. In the outgoing packets the NAT will change the private address, the source address, of the IP header to the corresponding external IP address, the global address.

In the same way the destination part of incoming messages is exchanged and sent out on the internal interface if it matches any of the external addresses in the table, thus the global address will be exchanged for a private address. These scenarios are shown in Figure 4.13.



*Figure 4.13: Address Translation*

There are a few problems that can occur when using NAT. One problem that can occur is, if a session is not initiated from the inside there will be no easy way to set up the translation table. The address to the right internal host must be found in some way. Another problem is, how the information is processed when application protocols like FTP put IP addresses inside the application data in the IP packets.

To be able to solve these kinds of problems, an Application Level Gateway (ALG) is needed to analyze these messages. In order to solve the first problem the internal clients must be registered at the ALG, which will give the ALG the internal address of the client. The second problem, which only applies to outbound messages, is also solved by having the ALG analyzing the message and exchanging the necessary parts. ALG will be described more, later in this document.

NAT comes in several flavors. They are called static NAT, dynamic NAT and Network Address Port Translation and Protocol Translation (NAPT-PT).

### **4.2.1 Static NAT**

Static NAT requires the same number of globally unique IP addresses, as there are hosts, in the private environment, that want to be able to connect to outside networks. As the name suggests, the mapping between local addresses and global addresses is intended to stay the same for a long period of time. The network administrator sets up the mapping between internal IP address and external IP address manually.

By using static NAT the problem with sessions, initiated from the outside with inbound requests, can be avoided. This is because of the one to one mapping in the translation table, i.e. when a new session is started from the outside, the NAT will always know which internal IP address to use.

### **4.2.2 Dynamic NAT**

With dynamic NAT, the external, i.e. global or public, IP addresses is collected into an IP address pool. With this flavor of NAT, it is not necessary to have the same number of external IP addresses as there are hosts wanting to connect to the outside. Reducing the number of external addresses that are needed is possible, since in dynamic NAT an assumption is made that all hosts do want to connect to the outside at the same time.

In the case when a host wants to make a connection to the outside, an external IP address is allocated from the pool of addresses managed by the NAT. For each allocated IP address, it reduces the number available and this can go on until all the IP addresses are allocated. The next host that tries to send outbound messages will not be able to send until a connection is closed down and the used IP addresses is returned to the pool of available IP addresses.

By looking in the TCP header, it's possible for TCP to verify if the packet closes down the connection, both the FIN bit (orderly release) and RST bit (abortive release) are considered [11]. With UDP there is no way to look at the UDP header and verify if a certain datagram will release the connection and return the address to the address pool, since UDP doesn't have these features.

The problem of when NAT can reclaim the address can be handled by using timeouts. When a mapping has not been accessed for a certain period of time the mapping is said to be dead, and the NAT reclaims the allocated address. TCP connections can be cut without any FIN message has been sent. TCP session may be terminated if it has not been used for 24 hours and for non-TCP connections, a timeout of a few minutes is proposed [11].

Dynamic NAT is more complex than static NAT, since we must keep track of communicating hosts and possibly even of connections, which requires looking at TCP information in packets.

For more detailed information concerning the NAT background readers are referred to the Appendix.

### **4.2.3 NAPT-PT**

A large number of users, offices and telecommuting employees have multiple network nodes in their office. These are running TCP and UDP applications, but have a single IP address assigned to their remote access router by their service provider to access remote networks. The increasing number of remote access users would be gained by NAPT-PT, which allows multiple nodes in a local network, at the same time access remote networks using the single IP address assigned to their router.

NAPT-PT is a special case of dynamic NAT and NAPT-PT uses port numbers as the basics for the address translation. With NAPT-PT, IPv6 nodes are allowed to communicate with the IPv4 nodes transparently using a single IPv4 address. The TCP and the UDP ports of the IPv6 nodes are translated into the TCP and the UDP ports of the registered IPv4 addresses. This allows the TCP and the UDP ports of a number of private hosts to be multiplexed into the TCP and the UDP ports of a single external address. A pool of external addresses is used when port translation is made under NAPT-PT. The number of ports available, that is  $2^{16} = 65536$ , limits the number of connections per IP address. Many ports are assigned to specific services and ports up to number 1024 are well known ports [17].

NAPT-PT translates the source IP address, source TCP port, source UDP port and related fields such as IP, TCP, UDP and ICMP header checksums, for packets outbound from the private network. The destination IP address, destination TCP port, destination UDP port and the IP, TCP and UDP header checksums are translated, for inbound packets [11].

NAPT-PT will solve the problem, which NAT would not be able to handle, when the pool of IPv4 addresses assigned for the translation is exhausted which implicates that newer IPv6 nodes will not be able to establish sessions with the outside world anymore [10].

#### **4.2.3.1 Address Translation**

Address translation is application independent and is often followed by application specific gateways, also known as ALG, to maintain application level transparency for translated sessions.

The translation of addresses has three phases: address binding, address lookup and translation, and address unbinding, which we describe in the following subsections.

##### **4.2.3.1.1 Address Binding**

Address binding is the phase where an IPv4 address is associated with an IPv6 address and vice versa and is fixed with static address assignments and is dynamic at session start up time with dynamic address assignments. New address bindings are made at the beginning of a new session. The same binding will be used for following sessions for session based packet translation.

With NAPT-PT many private addresses are mapped to a single globally unique address. The address binding is from the tuple of private address and private TCP and UDP port to the tuple of assigned address and assigned TCP and UDP port. When the first outgoing session is initiated by the tuple of private address and private TCP and UDP port on the private host, the binding is determined [11].

#### *4.2.3.1.2 Address Lookup and Translation*

Once a binding or address and TCP and UDP port is established it can be used for address lookup and translation. A datagram will forward the address or TCP and UDP port translation, for a datagram, from the origin address domain to the destination address domain with network addresses updated. The packets, which belong to the same session, will be subject to session lookup for translation aims [11].

#### *4.2.3.1.3 Address Unbinding*

The phase address unbinding is when a private address no longer is associated with a global address for translation aims, which is when the association between an IPv4 address and IPv6 address is broken.

The binding may be liquidated when the last session, which is based on an address or address and TCP and UDP port tuple, binding is liquidated [11].

#### **4.2.3.2 Protocol Translation**

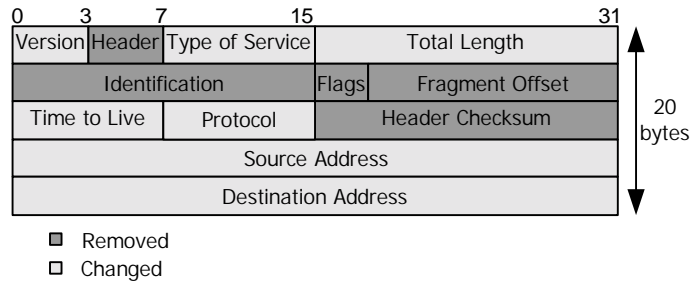
Protocol translation can be seen as simple mapping between two IP protocols, with some specific rules for handling fragments and path MTU discovery. The basic operation is to remove the original IP header and replace it with a new header from the other IP version.

Further we will describe an overview of the protocol translation process and the issues involved.

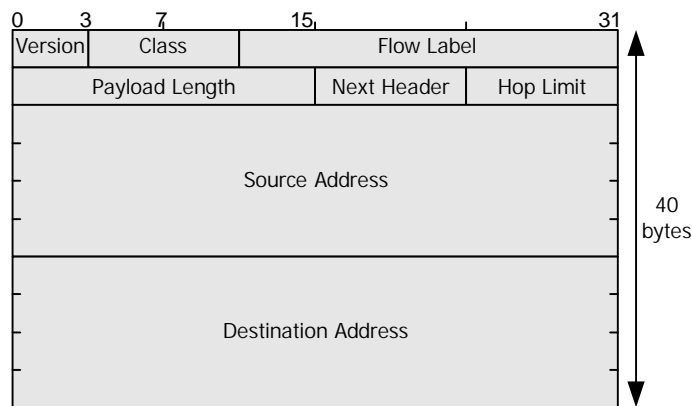
##### *4.2.3.2.1 IP Translation*

The headers in IPv6 and IPv4 have some similar fields, some fields that are either missing or have different sizes or meaning, as shown in Figure 4.14 and Figure 4.15. When translation is performed from one version of IP to the other version of IP, the translator either directly copies, translates, ignores, or sets fields in the IP header to a default value.

When translation is made from IPv6 to IPv4, the IPv4 checksum field is computed and when translation is reversed, that is for IPv4 to IPv6, the IPv4 checksum field is ignored. The IPv4 *total-length* field includes the IPv4 header size whereas the IPv6 *payload-length* field does not and the translation needs to account for this difference. The fields hop-limit and time-to-live (TTL) are copied and decreased by one. The protocol field can be instantly copied from one IP version to the other IP version, with ICMPv4 and ICMPv6 protocol numbers being the only exception [13].



*Figure4.14: IPv4 header format*



*Figure4.15: IPv6 header format*

With the exception of the IPv6 Fragment header, the translator silently ignores all other IPv6 extension headers and IPv4 options. The translator also ignores the IPv4 type-of-service and IPv6 traffic-class and flow-label fields, as there does not exist a semantic mapping between them. Since there is a direct mapping between the IPv4 and IPv6 fragmentation fields, the translation is straightforward when the translator receives a fragmented packet. There is a difference of the fragment identifier field between the two protocols. The Fragment identifier field in IPv6 has a wide of 32 bits and is twice as large as its IPv4 counterpart. When a translation is performed from IPv6 to IPv4, the lower 16 bits of the IPv6 fragmentation identifier is copied [13].

When the translator encounters an IPv4 packet, which is not fragmented and with the Don't Fragment flag set to false, that means fragmentation is allowed for that packet, it notes that by adding an IPv6 Fragment header and copying the IPv4 fragmentation fields to it, which states the following:

- The sender allows fragmentation and to ensure that packets are correctly reassembled, the fragmentation is carried end-to-end.

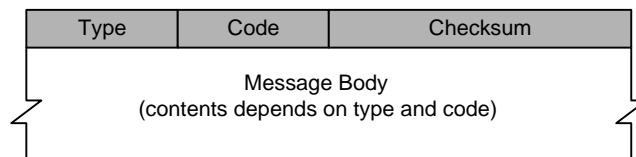
- Path MTU discovery is not used by the sender and the Don't Fragment bit must be set to false, should the packet be translated back to IPv4 [13].

The translation from IPv4 to IPv6 increases the packet size by at least 20 bytes due to the header length difference between the two protocols (28 bytes if it needs to add a Fragment header).

When the resulting packet is greater than the next-hop MTU and the Do not fragment flag is set, the translator will return an ICMP error message saying "Packet Too Big", otherwise the translator will fragment the resulting packet into next-hop MTU-sized packets [13]. In the case where the IPv4 host is sending MTU-sized packets e.g. such as NFS (Network File System), the fragmentation results in an inefficient packet stream. For this case we have the situation when an ICMPv4 "Packet Too Big" error message is returned to the IPv4 host, which contains a next-hop MTU that shows the size difference in the IP header size, giving the host the opportunity to readjust its path MTU value. Then if the host continues to send large packets, which does not support path MTU discovery, the translator will stop sending the ICMP error message and continue fragmenting the packet [5].

#### 4.2.3.2.2 ICMP Translation

The translator silently drops single hop ICMP messages and ICMP messages with unknown Type fields. The header format (Figure 4.16) is almost identical for ICMPv4 and ICMPv6 for the remaining messages, except for the ICMP Parameter Problem message.



*Figure 4.16: ICMP header format*

This message has an 8-bit pointer value in ICMPv4 and a 32-bit pointer value in ICMPv6. There is a counterpart in each version of ICMP messages and errors and these are: Echo Request, Echo Reply, Time Exceeded, Destination Unreachable, Packet Too Big, and Parameter Problem. Mostly there is a simple translation of the ICMP Type and Code fields. The Maximum Transmission Unit (MTU) field needs to be adjusted when a "Packet Too Big" error message reaches the translator during the translation to account for the difference between the IPv4 and the IPv6 header size [13]. The Pointer field also needs to be adjusted, for a Parameter Problem error message, to the corresponding field in the error causing IP header.

The Maximum Transmission Unit (MTU) field needs to be adjusted during the translation, when a Packet Too Big error message reaches the translator, to account for the difference between IPv4 and IPv6 header sizes. Also an adjustment has to be made in the Pointer field, for a Parameter Problem error message, to point to the corresponding field in the error causing IP header [13].

ICMP error messages, which include the IP header of the error-causing packet, have to be translated like a normal IP header that delivered the message [13]. It requires a recursive translation of the IP packet contained in the ICMP error message. The predetermined state is that the translation of the IP header is expected to change the length of the datagram, in which case the IPv6 Payload length and IPv4 Total-length fields also need to be adjusted. The translator at the end will silently drop all the IGMP messages [13].

#### **4.2.3.2.3 Adjusting Checksum Values**

Many higher layer protocols, for example TCP and UDP, compute their checksum values on a pseudo-header that consists of fields from the IP header. The checksum value needs to be adjusted with the difference between the original IP addresses and the translated IP addresses. ICMPv6, unlike ICMPv4, has a pseudo-header checksum just like UDP and TCP. The incremental checksum adjustment is calculated for ICMP Echo and Echo Reply informational messages, as only the Type value changes [13].

For ICMP Echo and Echo Request informational messages we calculate the incremental checksum adjustment, as only the Type value changes [13].

When translation is made from ICMPv6 to ICMPv4, a subtraction of the pseudo-header checksum has to be done. On the contrary, when translation is made from ICMPv4 to ICMPv6 an addition has to be made to the pseudo-header checksum. These informational messages can be fragmented either by the sending host or intermediate routers if their size exceeds the path MTU. In this case, the translator can't calculate the correct checksum value for ICMP Echo and Echo Request messages, since it doesn't know the total size of the packet, which it requires to add/subtract the pseudo-header checksum value when translating between the ICMP versions [13].

#### **4.2.3.3 ALG (Application Level Gateway)**

Application Level Gateway (ALG) is an application specific translation agent and for an application on a host in one address domain connecting to its counterpart running on a host on a different domain transparently, an ALG have to be used. To be able to get the application running across different address domains the ALG works together with the NAT-PT functionality in the Transition Box.

An ALG can work together with NAT-PT to set up state, use NAT-PT state information and modify application specific payload. ALGs make it easier for application specific communication between clients and servers [11].

To perform translations, ALGs are necessary for applications, which do not work transparently with NAPT-PT.

#### *4.2.3.3.1 DNS-ALG*

The Domain Name System (DNS) is a distributed database that maps Internet domain names to IP addresses. When DNS packets traverse from one address domain into another address domain, the Transition Box, which includes the DNS-ALG functionality changes the payload transparently.

In the DNS, IPv4 name to address mappings are held with “A” records and for IPv6 the address mappings are held with “AAAA” or “A6” records. In the case when an IPv4 node sends a name lookup request for an IPv6 node, the query is intended for the DNS server on the IPv6 network. This request would go through the Transition Box, and the DNS-ALG on the NAPT-PT would change the query type from an “A” record to an “AAAA” or “A6” record. The string “IN-ADDR.ARPA” would be changed with the string “IP6.INT”. Then as the DNS reply traverses from the DNS server from IPv6 to IPv4, the DNS-ALG changes the DNS replies from “AAAA” or “A6” records into “A” records. The IPv6 address also is exchanged with the IPv4 address assigned by the NAPT-PT. The DNS-ALG holds the mapping between IPv6 addresses and IPv4 addresses in the NAPT-PT. This implicates that the IPv4 node can contact the IPv6 node.

NAPT-PT without the support of DNS-ALG, provides one way connectivity between an IPv6 site and the IPv4 world meaning that only sessions initialized by IPv6 nodes internal to the IPv6 site can be translated, while sessions initiated by IPv4 nodes are dropped. This makes NAPT-PT a useful tool to IPv6 only site that need to be able to maintain connectivity with the IPv4 world without the need to organize servers visible to the IPv4 world.

In the case when an IPv6 node wants to communicate with an IPv4 node, a PREFIX (PREFIX::/96) has to be used in front of the IPv4 address of the IPv4 node. To start with, the IPv6 node must make a name lookup for the IPv4 node. This request would go through the Transition Box, and the DNS-ALG on the NAPT-PT would add an “A” record to an already existing “AAAA” or “A6” record. When the DNS-ALG receives the reply from the IPv4 node, it translates the reply adding the appropriate PREFIX to the IPv4 address and changes the DNS replies from “A” into “AAAA” or “A6” records. Now an IPv6 node can communicate with an IPv4 node [10].

#### *4.2.3.3.1.1 DNSSEC*

The Domain Name System Security Protocol (DNSSEC) is a method for supplying cryptographic verification information along with DNS messages. DNSSEC prevents clients from trusting false information. The DNSSEC offers three services [25]:

- Key distribution
  - The definition of a resource record format is to associate keys with DNS names and this allows the DNS to be used as a public

key distribution mechanism in support of DNS security itself and other protocols.

- Data origin authentication
  - Authentication is offered by associating resource record sets in the DNS with cryptographic digital signatures. The retrieved resource records are protected by the data origin authentication, but it offers no protection for DNS requests or for message headers.
- Transaction and request authentication
  - With transaction authentication a resolver can be certain of receiving messages from the server it thinks it queried and that the response is from the query it sent. Authenticating requests serves no function in older DNS servers, but will be required in future requests since they will require authentication.

The DNS packet can be a TCP or UDP packet and the Transition Box with NAPT-PT functionality translates the IP and TCP/UDP headers of the DNS packet. Then the DNS-ALG makes the DNS payload changes.

If DNS packets are encrypted/authenticated for each DNSSEC, DNS-ALG will fail because it won't be able to perform payload modifications and DNS-ALG can't support secure DNS name servers in the private domain. When modifications are attempted, zone transfers between DNSSEC servers will not be accepted and DNS-ALG will break any modified, signed responses, which is the case for all public side queries of private nodes when the DNS server is on the private side. The DNS-ALG would be able to modify digitally signed records if it had access to the source authentication key, but DNSSEC is designed to avoid distribution of this key to be able to maintain source authenticity [31].

So the conclusion is that the DNS-ALG functionality in the Transition Box can't support DNSSEC when DNS packets are encrypted/authenticated since DNS-ALG won't be able to modify the payload.

#### *4.2.3.3.2 FTP-ALG*

The File Transfer Protocol (FTP) will have to be able to negotiate the network protocol that will be used for data transfer during the transition between IPv4 and IPv6, since FTP just offer the ability to communicate information on IPv4 data connections. The Transition Box with the FTP-ALG functionality has to offer application level transparency for FTP, for the reason that an FTP control session carries in its payload, the IP address and tcp-port information for the data session.

When FTP applications running on IPv4, an IPv4 address and a tcp-port is included in the FTP PORT command and in the PASV response. EPRT and EPSV commands, which can be used on an IPv4 or IPv6 node, are used to

eventually replace the use of PORT and PASV commands. The aim of the FTP-ALG in the Transition Box is to smooth the progress of transparent FTP between IPv4 and IPv6 nodes. The commands EPRT and EPSV might or might not be supported by the IPv4 host in its FTP application. The format for EPRT is:  
***EPRT , net-prt , net-addr , tcp-port ,***

When an IPv4 host initiates the FTP session and use PORT or PASV commands, the FTP-ALG will translate these commands into EPRT and EPSV commands and then forward these to the IPv6 node. The PORT command will be translated into an EPRT command by setting the protocol field to IPv6 and translating the IPv4 host address into its NAPT-PT assigned IPv6 address and the tcp-port will be translated as well. From an IPv4 node, the PASV command is translated into an EPSV command with the protocol field set to IPv6. Then the EPSV response is translated back to a PASV response before forwarding it to the IPv4 host.

If the case is that a FTP session created by an IPv4 host is using the commands EPRT and EPSV, the FTP-ALG will just translate the parameters to these commands, without modifying the EPRT and EPSV commands. The protocol number field will be translated from IPv4 to IPv6; the IPv4 address will be translated to its NAPT-PT assigned IPv6 address and the tcp-port will also be translated.

When the IPv6 host creates a FTP session, there are two cases. The first case is when the IPv4 host supports the commands EPRT and EPSV. In this case the FTP-ALG will only translate the ***net-prt, net-addr*** and the ***tcp-port*** from IPv6 to its NAPT-PT assigned IPv4 information.

In the second case the IPv4 host doesn't support the commands EPRT and EPSV, the FTP-ALG will translate these commands to the corresponding PORT and PASV commands to be able to forward to IPv4.

#### ***4.2.3.3.3 SIP-ALG***

The SIP-ALG is a specific case of an Application Level Gateway as described in [9], by which both IP addresses in the Session Initiation Protocol (SIP) message and in the Session Description Protocol (SDP) body are statically mapped from one group to another. Consequently using SIP-ALG devices in NAPT-PT scenario requires that modifications be made to the SIP messages. These modifications are performed by the ALG [40]. In the following part of this section we present briefly SIP feature before we discuss SIP-ALG.

##### ***4.2.3.3.3.1 SIP Overview***

SIP (Session Initiation Protocol) is a text-based protocol that leverages the power of the Internet by borrowing common elements such as the format of HTTP, DNS, and email style addressing. SIP provides the necessary protocol elements to provide services such as personal mobility, call forwarding, calling and called party authentication.

The most fundamental SIP operation model is one in which two SIP user agents (UA) communicate directly. User agents may be LAN telephones or computer based end-user applications. Figure 4.7 shows a typical communication between two clients using SIP protocol

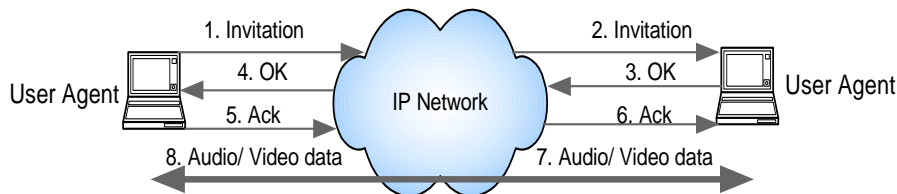


Figure 4.17: Communication using SIP

SIP is a general-purpose tool for the setup of multimedia sessions across the Internet. As a main part of its functionality, SIP must carry around the ports, IP addresses and domain names needed to describe the sessions it controls.

The protocol itself makes extensive use of network addresses located inside the message body, making it impossible to use SIP through basic network address translation without an Application Level Gateway (ALG) [40]. There are two issues in getting SIP to traverse NAT-PT. The first is getting SIP itself through, and the second is getting the media sessions it initiates through. The latter is by far the harder problem [41]. More about SIP and our SIP implementation will be discussed later in chapter 6.

### 4.2.3 NAT-PT Scenarios

Figure 4.18 shows how NAT-PT works. The scenarios that will be presented are communication between nodes in an IPv6 network with nodes in the IPv4 network and communication between nodes in an IPv4 network with nodes in the IPv6 network.

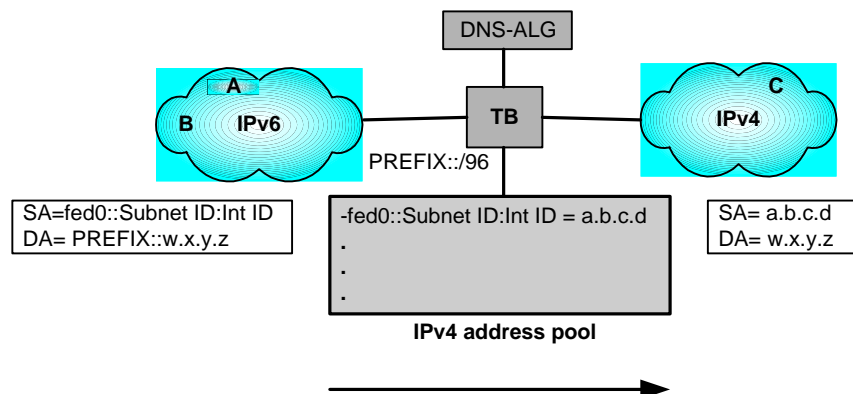


Figure 4.18: NAT-PT Scenario

There are two scenarios for NAT-PT:

- The first scenario is when node A in the IPv6 network wants to contact node C in the IPv4 network.
- The second scenario is when node C in the IPv4 network wants to contact node A in the IPv6 network.

Node A has an IPv6 source address: *fed0::Subnet ID:Int ID* and node C has an IPv4 source address: *w.x.y.z*.

#### *Scenario I*

1. When node A wants to contact node C, node A looks for the address for node C in its DNS. If the address couldn't be found the query is sent to the DNS-ALG.
2. The DNS-ALG ads an IPv4 query, thus an "A" record to an already existing IPv6 query, an "AAAA" record.
3. The Transition Box sends the query to the DNS in the IPv4 network.
4. Then node C's address, that is *w.x.y.z*, is returned to the DNS-ALG.
5. The DNS-ALG sends this reply to node A. Node A creates a packet with the IPv6 source address: *fed0::Subnet ID:Int ID* and the IPv4 destination address: *PREFIX::w.x.y.z*. To be able to get the destination address in IPv6 format, a *PREFIX::/96* must be added to the IPv4 address.
6. The packet will be sent to the Transition Box, and NAPT-PT will translate the source address with an address in the IPv4 address pool, in this case *a.b.c.d*. and also a port translation will be performed. NAPT-PT will remove the PREFIX part, to be able to get the destination address in IPv4 format. Now the packet has the source address: *a.b.c.d* and the destination address: *w.x.y.z*. Now node A in the IPv6 network is able to contact node C in the IPv4 network.

#### *Senario II*

In the second scenario, when node C in the IPv4 network wants to contact node A in the IPv6 network, node C looks for node A's address in the same way as the previous scenario. The change in this scenario is that the DNS-ALG returns *a.b.c.d*. as an address for node A. Node C creates an IPv4 packet with source address: *w.x.y.z* and destination address: *a.b.c.d*. This packet will be sent to the Transition Box, but the Transition Box doesn't know who the receiver is, since both node A and node B can have the address: *a.b.c.d*. This is a problem that NAPT-PT has, due to a limitation within NAPT-PT. The limitation, one server per service per IPv4 address, indicates that an IPv6 address only can keep one service to an IPv4 address. This leads to problems when using well-known ports, for example as SMS. Further limitation issues within NAPT-PT will be discussed in the next section of this chapter.

### **4.2.4 NAPT-PT limitations**

NAPT-PT has several limitations. First of all, limitations associated to NAT [16] are also associated to NAPT-PT. Here are the most important of them in detail, as well as some unique to NAPT-PT.

Like NAT, this is a one-way translation. It means that outside IPv6 domains, IPv4 application can't initiate communications with IPv6 hosts.

The protocol translator element of the NAPT-PT converts IP and ICMP headers between IPv4 and IPv6. Since the headers are largely unlike it is not possible to completely convert the semantic meaning of one header type to the other. Therefore not all the features specific to or supported by IPv6 can be translated by NAPT-PT [10].

Due to its characteristic, NAPT-PT operates as a network bridge. NAPT-PT has one interface to IPv6 and one interface to IPv4, and it does not use the routing functions to route packets. There is no usage of routing functions to route packets, the only possibility is that once a packet is in the IPv6 network it may be routed to further IPv6 subnets via a conventional router. The same is true for the IPv4 network.

Each IP session between an IPv6 host and an IPv4 host must be routed through the same TB. When a session begins, address translations for that session will be held within one TB only and not shared with any other TB.

It is not possible for the NAPT-PT router to support end-to-end security, such as IPsec. Since the NAPT-PT intercepts packets and rewrites them, any end-to-end authentication would immediately be invalidated. One possible method of overcoming this problem would be to make the NAPT-PT router a trusted host. A further problem would exist where packets are encrypted. Without the encryption keys it will not be possible for NAPT-PT to read and translate any of the IP packets.

## **4.2.5 Alternative methods to NAPT-PT**

### **4.2.5.1 RSIP**

The Realm Specific IP (RSIP) allows a host from one addressing domain in another addressing domain by permitting it to use resources, such as addresses and other routing parameters, from the second addressing domain. The RSIP doesn't require an ALG for communication between an RSIP host and a host in a different addressing domain.

RSIP can improve some IP address shortage problems in some scenarios, but it isn't a long-term solution to the IP address shortage problem. RSIP does allow end-to-end packet transparency between two different addressing domains but it may not be transparent to all applications [42].

### **4.2.5.2 BIS**

The Bump-In-The-Stack (BIS) allows a non-IPv6 application on an IPv4 host to communicate with IPv6 only hosts. The concept of BIS is when an IPv4 application needs to communicate with an IPv6 only host the IPv6 address of that host is mapped into an IPv4 address out of a pool local to the dual stack

hosts. The IPv4 packet generated for the communication is translated into an IPv6 packet. This requires implementation of BIS on IPv4/IPv6 and IPv4 hosts.

So the BIS method allows IPv4 only applications on a dual stack host to communicate with IPv6 only hosts [22].

#### **4.2.5.3 SOCKS64**

SOCKS64 is a gateway system that allows improved SOCKS connections from IPv4 hosts and relays it to IPv4 or IPv6 hosts. For “socksified” sites, which use SOCKS aware clients and SOCKS servers, the SOCKS64 offers an easy way to let IPv4 hosts connect to IPv6 hosts. No packet translation or DNS modification are needed. This method also allows IPv6 hosts to connect to IPv4 hosts, IPv4 hosts over IPv6 networks and IPv6 hosts over IPv4 networks [43]. The disadvantage with SOCKS64 is that it requires that all client nodes have to be SOCKS enabled [52].

#### **4.2.6 Summarizing NAPT-PT**

NAPT-PT makes it possible for the communication between IPv6 nodes and IPv4 nodes transparently by using a single IP address and NAPT-PT use port numbers as the basics for the address translation.

A problem that NAPT-PT can solve and NAT can't deal with is when the pool of IPv4 addresses assigned for the translation is exhausted, this implicates that newer IPv6 nodes will not be able to establish sessions with the outside world anymore [10].

The disadvantage with NAPT-PT is when a node from IPv4 network wants to communicate with a node from IPv6 network, due to the limitation of one server per service per IPv4 address; which indicates that an IPv6 address only can keep one service to an IPv4 address.

When the number of IPv4 addresses was reaching an end, NAT was used as a solution for the IPv4 address limitation despite the disadvantages with NAT. Later IPv6 and the dual-stack mechanism were introduced because of the limitation of IPv4 addresses and drawbacks with NAT. Furthermore the expansion of new devices such as the PDA, as mentioned above, leads to that the dual-stack mechanism will not be valid and therefore it will lead to IPv6 only sites. So the conclusion is that the NAPT-PT functionality has to be used during the transition period from IPv4 to IPv6.

## **5 Experimental Setup**

We participate to set up a lab network to test the implementation for the Transition Box with the different 6to4 and NAPT-PT scenarios and to verify the functionality of our implementation of SIP-ALG (Session Initiation Protocol-Application Level Gateway).

## 5.1 Lab Network

This description of the lab network in the Transition Box thesis is intended to give an overview of the lab network.

It will detail which hardware and software that have been used, how the software and operating system (OS) have been configured. Relevant details of configuration files can be found at Appendix E.

## 5.2 Hardware and software used

For the lab network, five Dell Optiplex GX110 with a Pentium III 733 MHz and 128 MB RAM were used. They were equipped with a D-Link Quad Channel Server Card DFE-570TX (4 ports at 100 Mbit/s).

All machines run FreeBSD 4.1.1-RELEASE, which include Berkeley Packet Filter, recompiled with KAME IPv6 stack and 6to4 support. For NAPT-PT functionality, two machines run the NAPT-PT code written by British Telecom, amended to compile on these. The IPv6 hosts have been updated with BIND9 (Berkeley implementation of name resolvers and servers version 9).

## 5.3 Configuration and Setup

The lab setup, to test 6to4 functionality, is illustrated in Figure 5.1.



*Figure 5.1: 6to4 Test setup*

Here, the hosts Venus and Mars act as 6to4 routers interconnecting the two IPv4 clouds without explicit tunneling.

The prefix, 2002:/16 is used for 6to4, and since Venus has an IPv4 address of 10.115.32.3, translating into hexadecimal 0a732003, the left domain gets a network address of 2002:a73:2003::/64. Since mars has an IPv4 address of 10.115.32.2, the right IPv6 cloud will get a prefix of 2002:a73:2002::/64.

The physical ethernet interfaces are denoted dcX. The 6to4 pseudo-interface is denoted stf0 in both of Venus and mars. The end hosts, i.e. mercury and Saturn do not require any special configuration and does normal IPv6 auto configuration to form their full IPv6 address including prefix.

To test the NAPT-PT, the lab was set up according to Figure 5.2.

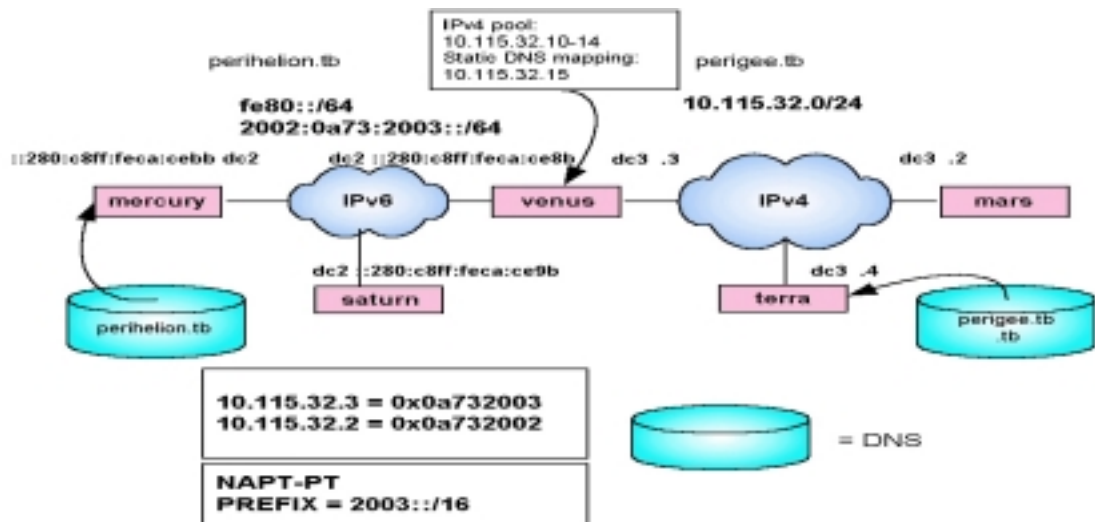


Figure 5.2: The Lab setup

In this setup Venus runs the NAPT-PT software to translate packets between the IPv6 and IPv4 clouds. Venus holds a static mapping between an IPv4 address and an IPv6 address for the DNS in mercury. The DNS configuration will be further described below. It also manages an IPv4 address pool, i.e. 10.115.32.10-14. Hosts inside the IPv6 cloud use a special prefix, i.e. 2003:/16, to communicate with IPv4 hosts through the NAPT-PT box. They form the full address by attaching the IPv4 address to the prefix, e.g. 2003::0a73:2004 for terra (at IPv4 address 10.115.32.4).

While the hosts Saturn and mars are pure IPv6 and IPv4 hosts respectively, mercury and terra act as Domain Name Servers (DNSs).

Mercury is the authoritative name server for the domain perihelion.tb and maps only IPv6 addresses in that domain. Terra is the authoritative name server for the perigee.tb domain as well as acting top domain server for the .tb domain. It delegates the perihelion.tb domain to the IPv4 address 10.115.32.15, which in Venus maps to mercury's IPv6 address. DNS traffic traversing Venus is intercepted by the DNS-ALG and translated as applicable. Mercury redirects DNS queries to other domains than it's own to terra via its IPv6 address formed with the NAPT prefix, as shown above.

For more detailed information concerning the configuration readers are referred to the configuration files found at Appendix E.

In the following section we will briefly present the software components that used in our test machines, these are FreeBSD, Berkeley Packet Filter and KAME IPv6 stack. Then we will describe how to enable the 6to4 mechanism in FreeBSD.

## **5.5 FreeBSD**

FreeBSD is an UNIX-like operating system for the i386 and Alpha/AXP platforms based on U.C. Berkeley's 4.4BSD-Lite release. It is also based indirectly on William Jolitz's port of U.C. Berkeley's Net/2 to the i386, known as 386BSD, though very little of the 386BSD code remains.

FreeBSD is a multi-user system; several people can access the same computer at the same time. This includes more than the "File Serving" capabilities of WinNT and Novell Servers. FreeBSD also gives the user much more control over the system than DOS or Windows 2000. The user has the ability to modify system parameters "live", not just edit the configuration file and have the changes take affect after he have rebooted. For example, the user can change the IP address of his machine and then test immediately to see if it is working. He doesn't have to wait 5 minutes for his computer to reboot to see if a change has worked.

A fuller description of what FreeBSD is and how it can work may be found on the FreeBSD official site <http://www.freebsd.org> [65].

### **5.5.1 KAME IPv6 stack**

The KAME Project is a joint effort of people from seven companies in Japan to provide a free IPv6/IPsec stack for BSD variants. KAME provides a free IPv6 platform for research/commercial use.

KAME Project aims to provide free reference implementations of

- IPv6
- IPsec (for both IPv4 and IPv6)
- Advanced internetworking such as ATM, mobility

Currently KAME is being developed for several Free Software BSD variants including FreeBSD, NetBSD, OpenBSD, and also for the commercial BSDI OS. This new release is available in the following URL: <http://www.kame.net/snap-users/>.

A detail description of what KAME is and how it can work may be found on this site <http://www.kame.net> [66].

### **5.5.2 The Berkeley Packet Filter**

Berkeley Packet Filter (BPF) is an efficient and portable interface for network monitoring. Many version of Unix provide facilities for user-level packet capture, making possible the use of general purpose workstations for network monitoring. Because network monitors run as user-level processes, packets must

be copied across the kernel/user-space protection boundary. This copying can be minimized by deploying a kernel agent called packet filter, which discards unwanted packets as early as possible.

The BSD packet filter originated in the Xerox Alto. Packet filters are written in a simple stack-based language and inserted into the running kernel. For each incoming data packet, the filters are executed until one application accepts the packet. There are no branch instructions and only a rudimentary set of operations. Words at constant offsets in the packet can be examined using comparisons and the three boolean operators AND, OR, and XOR. No other arithmetic operations are available [60].

The two main components in BPF is the network tap and the packet filter. The network tap collects copies of packets from the network, device drivers and delivers them to listening applications. The filter decides if a packet should be accepted and, if so, how much of it should be copied to the listening application.

### 5.5.3 Enabling 6to4 in FreeBSD

FreeBSD has a special pseudo-device that can be used to set up 6to4 called **stf**. We put pseudo-device **stf** in our kernel configuration. After that we have a kernel set up for both IPv4 and IPv6, and we have **stf0** available, and our IPv4 configuration is set up, we add the following to our rc.conf file:

```
ipv6_enable="YES"  
ipv6_network_interfaces="auto"  
ipv6_gateway_enable="YES"  
ipv6_prefix_nn0="2002:xxxx:xxxx"  
stf_interface_ipv4addr="xxx.xxx.xxx.xxx"
```

Replacing the xs with machine's IPv4 address, and nn0 with interface's name. More about the configuration file that we use can be founded in Appendix D.

This setup presumes that we have a static IPv4 address. It is possible to use 6to4 with a dynamic address, but this means that our IPv6 prefix will change every time the IPv4 address does.

Having done this, we can now exchange packets with anyone else using 6to4 anywhere on the net. But what about people not using 6to4 - sites on the 6bone or in other IPv6 address spaces? To get to non-6to4 addresses, we need to use a relay router. That is a machine that is set up both for 6to4 and a connection to some other address space. If we set such a machine as our default route, it will pass our packets on to the rest of the IPv6 universe.

We should always pick the entry in this list closest to us. We should measure the distance with an IPv4 traceroute (IPv6 traceroute will be pointless, as it will always be one hop). We should also try and select higher bandwidth hosts, as low bandwidths will represent a bottleneck for our outgoing non-6to4 traffic. Once we have selected a relay router, make it our IPv6 default route by adding this to our rc.conf:

```
ipv6_static_routes="default"  
ipv6_route_default="default relay_router"
```

For simplicity reason we don't use relay route in our test.

## 6 SIP-ALG implementation

In this chapter we briefly present how our SIP-ALG implementation can be used in NAPT-PT to translate SIP messages. This particular part (SIP-ALG) of NAPT-PT implementation has been developed for a router running the FreeBSD operating system and using the KAME IPv6 stack.

When an IP packet arrives to the TB it will be examined to determine whether it is a tunneled packet or not. If not, the NAPT-PT receives the packet and consequently the main program in NAPT-PT (Nat\_pt.c) will start to create two main network interfaces to the IPv4 and IPv6 networks. These received packets that wish to cross the IPv4/IPv6 border. Three other network interfaces are also created, specifically to receive IPv4 ARP request and reply, and IPv6 Neighbor Advertisement packets. After that the ALG manager determines whether a packet must be sent to an ALG for further processing. An ALG is necessary where IP address information is contained within the packet payload and thus requires specific translation. The current version of NAPT-PT is limited in its ALG support, namely DNS-, FTP- and SIP-ALG. Extra ALG's may be appended to the implementation.

SIP Application Level Gateway receives an IP packet from the ALG manager and translates SIP messages from IPv6 to IPv4 and vice versa. If the message is one of the following commands (INVITE, BYE, CANCEL, ACK, REGISTER, 200, 404) a translation is required. If not the packet is returned unchanged. The translation will be done by the functions TRANSLATE\_REQUEST and TRANSLATE\_RESPONSE depending on whether it is a request or response message the SIP-ALG retrieve. These functions translate the header of the SIP message according to the SIP URL syntax specification (see Appendix c) with the following additions for IPv6:

```
Host = hostname | IPv4address | IPv6address  
IPv6address = “[ 1*digit (“:” | “.”) 1*digit “]”
```

The syntax of our IPv6 address is beginning with “[“ and end with “]”, this to separate IPv6 address from port number. According to the SIP URL syntax, which is designed for IPv4 addresses, the port number can be separated from the IPv4 address by adding “:”, this is possible for IPv4 but not for IPv6 address.

After translation of the message a new UDP checksum will be calculated by the function (ALG\_Manager\_Calculate\_Checksum\_UDP) and the new packet will be sent, Figure 6:1 demonstrate the flowchart of our SIP-ALG.

Note that this SIP-ALG is currently only discussing an implementation for UDP SIP traffic, and that the modification required for supporting TCP are currently unknown [40].

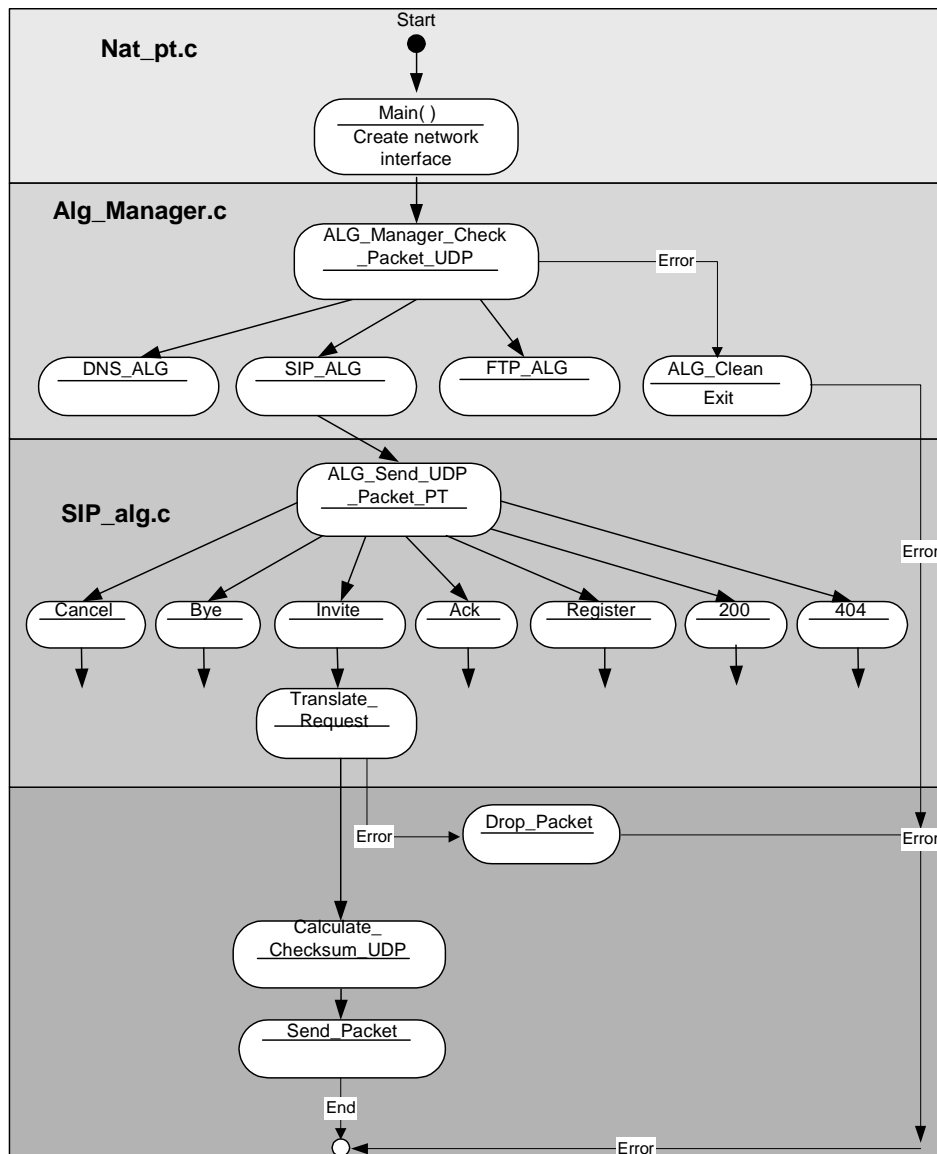


Figure 6.1: The SIP-ALG flowchart

## 7 Model Limitation and future work

Several limitations have been mentioned in this report. For 6to4 tunneling, encapsulation adds an additional load to the network. Communicating sites need to implement 6to4 to recognize the TLA. Despite of these facts 6to4 is one of the simplest and most widely used tunneling mechanism.

Unlike 6to4, NAPT-PT has several limitations. Earlier in this document we mentioned some of them. NAPT-PT is one-way translation. It means that outside the IPv6 domains, IPv4 applications can't initiate communications with IPv6 hosts.

All received packets by NAPT-PT have to be examined and sometimes translated. This can lead to more delay in the network. Further limitation is that for example, requests and responses pertaining to a session must be routed via the same NAPT-PT, as the NAPT-PT maintains state information for sessions established through it. For this reason, it is often suggested that NAPT-PT operates on a border router unique to a stub domain, where all IP packets are either originated from the domain or destined to the domain. However, such a configuration would turn the NAPT-PT into a single point of failure.

Thus a possible future work can be to obstruct these drawbacks. In addition there are still a lot of issues to deal with concerning SIP-ALG. For example there is no standard SIP syntax for IPv6, we had our syntax as described before in SIP-ALG implementation's section. Furthermore, SIP-ALG is currently only discussing an implementation for UDP SIP traffic, and the modifications required for supporting TCP are currently unknown. Also this ALG only describes modifications for implementations, which use SDP to describe the media stream [40].

## **8 Conclusions**

The goals of this master's thesis, outlined in section 1.3, have been achieved. The first one was to study a transition mechanism, which include the functionalities NAPT-PT and 6to4. These will provide a way to interconnect IPv6 with IPv4 domains, and IPv6 domains via IPv4 domains. We have also described alternative methods. Finally an implementation of SIP-ALG based on FreeBSD has been made.

The conclusion is that the Transition Box, including the functionalities NAPT-PT and 6to4, has to be used during the transition period from IPv4 to IPv6 until we reach the point that we move to IPv6 network. This mechanism can cooperate with a number of applications, such as FTP, DNS and SIP. We made an implementation of SIP-ALG, which support the SIP application traffic through Transition Box. There is, however still work to do in this area.

## 9 Acronyms

ALG	Application Level Gateway
API	Application Programming Interface
BPF	Berkeley Packet Filter
BGP	Border Gateway Protocol
BIND9	Berkeley implementation of name resolve and servers version 9
BSD	Berkeley Software Distribution
FreeBSD	Free Berkeley Software Distribution
NetBSD	Net Berkeley Software Distribution
OpenBSD	Open Berkeley Software Distribution
CIDR	Classless Inter Domain Routing
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Services
DNSSEC	Domain Name System Security Protocol
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force <sup>1</sup>
IP	Internet Protocol
IPsec	Secure Internet Protocol
Ipng	Internet Protocol next generation
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISP	Internet Service Provider
KAME	is abbreviation for " <b>K</b> Arigo <b>M</b> E" <sup>2</sup>
LAN	Local Area Network
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NAPT-PT	Network Address Port Translation and Protocol Translation
ND	Neighbor Discovery
PT	Protocol Translation
RFC	Request For Comments
SDP	Session Description Protocol
SIIT	Stateless IP/ICMP Translator
SIP	Session Initiation Protocol
TB	Transition Box
TCP	Transport Control Protocol
TLA	Top Level Aggregator
TTL	Time To Live
UA	User Agent
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator

<sup>1</sup> IPng Transition working Group

<sup>2</sup> KArigoME means "turtle" in Japanese

3G                    Third Generation  
3GPP                Third Generation Partnership Project

## 10 Terminology

*The following terms are used in this document:*

### **Types of Nodes**

**IPv4-only node:** A host or router that implements only IPv4. An IPv4-only node does not understand IPv6. The installed base of IPv4 hosts and routers existing before the transition begins are IPv4-only nodes.

**IPv6/IPv4 node:** A host or router that implements both IPv4 and IPv6.

**IPv6-only node:** A host or router that implements IPv6, and does not implement IPv4. The operation of IPv6-only nodes is not addressed here.

**IPv6 node:** Any host or router that implements IPv6. IPv6/IPv4 and IPv6-only nodes are both IPv6 nodes.

**IPv4 node:** Any host or router that implements IPv4. IPv6/IPv4 and IPv4-only nodes are both IPv4 nodes.

### **Types of Addresses**

**Internet Address:** A four octet (32 bit) source or destination address consisting of a Network field and a Local Address field.

**IPv4 Global Address:** An IPv4 address that is globally routable on the Internet.

**Local Address:** the address of a host within a network. The actual mapping of an Internet local address on to the host addresses in a network is quite general, allowing for many to one mapping.

**IPv4-compatible IPv6 address:** An IPv6 address bearing the high-order 96-bit prefix 0:0:0:0:0:0, and an IPv4 address in the low-order 32-bits. IPv4-compatible addresses are used by IPv6/IPv4 nodes, which perform automatic tunneling.

**IPv6-native address:** The remainder of the IPv6 address space. An IPv6 address that bears a prefix other than 0:0:0:0:0:0.

### **Techniques Used in the Transition**

**IPv6-over-IPv4 tunneling:** The technique of encapsulating IPv6 packets within IPv4 so that they can be carried across IPv4 routing infrastructures.

**Configured tunneling:** IPv6-over-IPv4 tunneling where the IPv4 tunnel endpoint address is determined by configuration information on the encapsulating node. The tunnels can be either unidirectional or bi-directional. Bi-directional configured tunnels behave as virtual point-to-point links.

Automatic tunneling: IPv6-over-IPv4 tunneling where the IPv4 tunnel endpoint address is determined from the IPv4 address embedded in the IPv4 compatible destination address of the IPv6 packet being tunneled.

IPv4 multicast tunneling: IPv6-over-IPv4 tunneling where the IPv4 tunnel endpoint address is determined using Neighbor Discovery [7]. Unlike configured tunneling this does not require any address configuration and unlike automatic tunneling it does not require the use of IPv4-compatible addresses. However, the mechanism assumes that the IPv4 infrastructure supports IPv4 multicast.

**Modes of operation of IPv6/IPv4 nodes**

IPv6-only operation: An IPv6/IPv4 node with its IPv6 stack enabled and its IPv4 stack disabled.

IPv4-only operation: An IPv6/IPv4 node with its IPv4 stack enabled and its IPv6 stack disabled.

IPv6/IPv4 operation: An IPv6/IPv4 node with both stacks enabled.

DTI: Tunneling Interface. An Interface, encapsulates IPv4 packets into IPv6 packets.

**6to4**

6to4 Tunneling technique between IPv6 and IPv4

6to4 pseudo-interface: 6to4 encapsulation of IPv6 packets inside IPv4 packets occurs at a point that is logically equivalent to an IPv6 interface, with the link layer being the IPv4 unicast network. This point is referred to as a pseudo-interface. Some implementors may treat it exactly like any other interface and others may treat it like a tunnel end-point.

6to4 prefix: An IPv6 prefix constructed according to the rule in chapter 2.

6to4 address: An IPv6 address constructed using a 6to4 prefix.

Native IPv6 address: An IPv6 address constructed using another type of prefix than 6to4.

6to4 router: An IPv6 router supporting a 6to4 pseudo interface. It is normally the border router between an IPv6 site and a wide-area IPv4 network.

6to4 host: An IPv6 host which happens to have at least one 6to4 address. In all other respects it is a standard IPv6 host. Note: an IPv6 node may in some cases use a 6to4 address for a

configured tunnel. Such a node may function as an IPv6 host using a 6to4 address on its configured tunnel interface, and it may also serve as an IPv6 router for other hosts via a 6to4 pseudo-interface, but these are distinct functions.

**6to4 site:** A site running IPv6 internally using 6to4 addresses, therefore containing at least one 6to4 host and at least one 6to4 router.

**Relay router:** A 6to4 router configured to support transit routing between 6to4 addresses and native IPv6 addresses.

**6to4 exterior routing domain:** A routing domain interconnecting a set of 6to4 routers and relay routers. It is distinct from an IPv6 site's interior routing domain, and distinct from all native IPv6 exterior routing domains.

**Other**

**Protocol:** In this document, the next higher level protocol identifier, an Internet header field.

**TCP:** Transmission Control Protocol: A host-to-host protocol for reliable communication in Internet environments.

**Time to Live:** An Internet header field which indicates the upper bound on how long this internet datagram may exist.

**UDP:** User Datagram Protocol: A user level protocol for transaction

**6over4:** Tunneling technique between IPv6 and IPv4

**6Bone:** The 6bone is an experimental world scale IPv6 network, where IPv6 applications connectability is tested as well as wither IPv6 will really work in actual life like situations. The World 6bone consists of a group of regional 6bones.

**ICMPv4:**

**ICMPv6:**

**FreeBSD:**

**SHASTA:**

**KAME:**

An IPv6/IPsec implementation of IPv6 working group, 6bone project in Japan. Its goals are: to provide FREE IPv6 platform for research/commercial use, to provide FREE IPsec to the world, and to provide the FREE reference code for Internetworking for the 21st century. KAME kit is formerly called "Hydrangea" IPv6/IPsec kit.

## 11 Reference

### Books

- [1] Christian Huitema, *IPv6: THE NEW INTERNET PROTOCOL*, Prentice Hall, 1996.
- [2] Stephen A. Thomas, *IPng and the TCP/IP Protocols*, Wiley Computer Publishing, 1996.
- [3] W. Richard Stevens, *TCP/IP Illustrated Volume 1 – The Protocols*, Addison Wesley Longman, 1999.
- [4] W. Richard Stevens, *UNIX Network Programming*, Prentice Hall, 1998.

### RFC (<http://www.ietf.org>)

- [5] RFC: 1191, *Path MTU Discovery*.
- [6] RFC: 1296, *Internet Growth (1981-1991)*.
- [7] RFC: 2374, *An IPv6 Aggregatable Global Unicast Address Format*.
- [8] RFC: 2461, *Neighbor Discovery for IP Version 6 (IPv6)*.
- [9] RFC: 2543, *SIP: Session Initiation Protocol*.
- [10] RFC: 2766, *Network Address Translation – Protocol Translation (NAT-PT)*.
- [11] RFC: 2663, *IP Network Address Translator (NAT) Terminology and Considerations*.
- [12] RFC: 2428, *FTP Extensions for Ipv6 and NATs*.
- [13] RFC: 2765, *Stateless IP/ICMP Translation Algorithm (SIIT)*.
- [14] RFC: 1191, *Path MTU Discovery*.
- [15] RFC: 2694, *DNS extensions to Network Address Translators (DSN\_ALG)*.
- [16] RFC: 1631, *The IP Network Address Translator (NAT)*.
- [17] RFC: 1700, *ASSIGNED NUMBERS*.
- [18] RFC: 1715, *The H Ratio for Address Assignment Efficiency*.
- [19] RFC: 1933, *Transition Mechanisms for IPv6 Hosts and Routers*.
- [20] RFC: 2893, *Transition Mechanisms for IPv6 Hosts and Routers*.
- [21] RFC: 2529, *IPv6 over IPv4 without Explicit Tunnels*.
- [22] RFC: 2767, *Dual Stack Hosts using the "Bump-In-The-Stack" Technique (BIS)*.
- [23] RFC: 791, *Internet Protocol*.
- [24] RFC: 2460, *Internet Protocol Version 6 (IPv6) Specification*.
- [25] RFC: 2535, *Domain Name System Security Extensions*.

### Internet Draft (<http://www.ietf.org>)

- [30] draft-iab-nat-implications-09.txt, *Architectural Implications of NAT*.
- [31] draft-ietf-nat-protocol-complications-04.txt, *Protocol Complications with IP NAT*.
- [32] draft-ietf-ngtrans-6to4-07.txt, *Connection of IPv6 Domains via IPv4 Clouds*.
- [33] draft-ietf-ngtrans-introduction-to-ipv6-transition-04.txt, *On overview of the introduction of IPv6 in the Internet*.
- [34] draft-ietf-ngtrans-translator-03.txt, *Overview of Transition Techniques*.

- [35] draft-ietf-ngtrans-dstm-02.txt, *Dual Stack Transition Mechanism (DSTM)*.
- [36] draft-ietf-iab-case-for-ipv6-06.txt, *The Case for IPv6*.
- [37] draft-ietf-ipngwg-default-addr-select-00.txt, *Default Address Selection for IPv6*.
- [38] draft-ietf-ngtrans-mech-06.txt, *Transition Mechanisms for IPv6 Hosts and Routers*.
- [39] draft-ietf-ngtrans-broker-06.txt, *Tunnel broker*.
- [40] draft-biggs-sip-nat-01.txt, *A SIP Application Gateway for Network Address Translation*.
- [41] draft-rosenberg-sip-firewalls-00.txt
- [42] draft-ietf-nat-rsip-framework-05.txt, *Realm Specific IP*.
- [43] draft-ietf-ngtrans-socks-gateway-05.txt, *A SOCKS-based IPv6/IPv4 Gateway Mechanism*.

**Reports and articles**

- [45] Elwyn Davies, Tim Roberts, *Next Generation IP Network*, A White Paper
- [46] David C. Lee, Daniel L. Lough, *The Internet Protocol version 6*, IEEE Potentials, 1998.
- [47] Jun-ichiro Itoh, *Constructing IPv6 Protocol Stack on Apertos*.
- [48] Hossam Afifi, Laurent Toutain, *Methods for IPv4-IPv6 Transition*, Computer and Communications, 1999.
- [49] David C. Lee, Daniel L. Lough, *Aspects of the Internet Protocol version 6*, IEEE Networks, January 1998.
- [50] William Stallings, *In the future with IPv6*, Dr. Dobb's Journal, August 1998.
- [51] Brian E. Carpenter, Keith Moore, Bob Fink, *Connecting IPv6 Routing Domains Over the IPv4 Internet*, The Internet Protocol Journal, March 2000.
- [52] Pete Loshin, *IPv6 Over Everything*, Data Communications, October 1999.
- [53] G. Koprowski, *Limitations of IPv4 Ensure Future of IPv6*, Computer, December 1998.
- [54] Hinden -RM, *IP Next Generation Overview*, Communications-of-the-ACM, June 1996.
- [55] Hosny-W, Kamel-T, Shaheen-S, *A comparative analysis of transition mechanisms for IPv6/IPv4 routing environment*, 5th International Computer Science Conference ICSC'99, 1999.
- [56] IETF IPv6 Transition Working Group (ngtrans) information, including status of all its current projects, can be found at: <http://www.6bone.net/ngtrans/>.
- [57] Brian Haberman, Nortel Networks Internet Protocol Version 6 Roadmap, October 2000.
- [58] La Monica, P., *Everybody's Talking about Voice stream*, Red Herring, July 2000, nr. 80, <http://www.redherring.com/investor/2000/0712/inv-voicestream071200.html>.
- [59] Charlie Perkins, *IPv6 from the Viewpoint of Mobile Wireless*. The COOK Report on Internet, Volume IX No.11, February 2001.

- [60] S. McCanne, and V. Jacobson, *The BSD Packet Filter: A New Architecture for User-level Packet Capture*. Proceedings of the 1993 Winter USENIX Technical Conference (San Diego, CA, Jan. 1993), USENIX.
- [61] Technical Report (TR) 23.821, *Architecture Principles for Release 2000*, <http://www.3gpp.org>.
- [70] Terry Boland, Tom Hussey, *Building the Foundation of Profitable 3G Wireless Internet and The Road to IPv6*, A White Paper

**Web site**

- [62] [http://www.isoc.org/inet2000/cdproceedings/1e/1e\\_1.htm](http://www.isoc.org/inet2000/cdproceedings/1e/1e_1.htm)
- [63] <http://www.isc.org/ds/hosts.html>
- [64] IETF IPv6 Working Group (ipngwg) information, can be found at: <http://www.ietf.org/html.charters/ipngwgcharter.html>
- [65] <http://www.freebsd.org>
- [66] <http://www.kame.net>.
- [67] <http://www.tekpress.com>
- [68] <http://www.3gpp.org>

## 12 Appendix A

### Complete call setup of a two party call using SIP

This section can also be found in RFC 2543.

For two-party Internet phone calls, the response must contain a description of where to send the data. In the example below, Bell calls Watson. Bell indicates that he can receive RTP audio codings 0 (PCMU), 3 (GSM), 4 (G.723) and 5 (DVI4).

```
C->S: INVITE sip:watson@boston.bell-tel.com SIP/2.0
Via: SIP/2.0/UDP kton.bell-tel.com
From: A. Bell
To: T. Watson
Call-ID: 3298420296@kton.bell-tel.com
CSeq: 1 INVITE
Subject: Mr. Watson, come here.
Content-Type: application/sdp
Content-Length: ...
```

```
v=0
o=bell 53655765 2353687637 IN IP4 128.3.4.5
s=Mr. Watson, come here.
c=IN IP4 kton.bell-tel.com
m=audio 3456 RTP/AVP 0 3 4 5
```

```
S->C: SIP/2.0 100 Trying
Via: SIP/2.0/UDP kton.bell-tel.com
From: A. Bell
To: T. Watson ;tag=37462311
Call-ID: 3298420296@kton.bell-tel.com
CSeq: 1 INVITE
Content-Length: 0
```

```
S->C: SIP/2.0 180 Ringing
Via: SIP/2.0/UDP kton.bell-tel.com
From: A. Bell
To: T. Watson ;tag=37462311
Call-ID: 3298420296@kton.bell-tel.com
CSeq: 1 INVITE
Content-Length: 0
```

```
S->C: SIP/2.0 182 Queued, 2 callers ahead
Via: SIP/2.0/UDP kton.bell-tel.com
From: A. Bell
To: T. Watson ;tag=37462311
Call-ID: 3298420296@kton.bell-tel.com
CSeq: 1 INVITE
```

Content-Length: 0  
S->C: SIP/2.0 182 Queued, 1 caller ahead  
Via: SIP/2.0/UDP kton.bell-tel.com  
From: A. Bell  
To: T. Watson ;tag=37462311  
Call-ID: 3298420296@kton.bell-tel.com  
CSeq: 1 INVITE  
Content-Length: 0

S->C: SIP/2.0 200 OK  
Via: SIP/2.0/UDP kton.bell-tel.com  
From: A. Bell  
To: ;tag=37462311  
Call-ID: 3298420296@kton.bell-tel.com  
CSeq: 1 INVITE  
Contact: sip:watson@boston.bell-tel.com  
Content-Type: application/sdp  
Content-Length: ...

v=0  
o=watson 4858949 4858949 IN IP4 192.1.2.3  
s=I'm on my way  
c=IN IP4 boston.bell-tel.com  
m=audio 5004 RTP/AVP 0 3

The example illustrates the use of informational status responses. Here, the reception of the call is confirmed immediately (100), then, possibly after some database mapping delay, the call rings (180) and is then queued, with periodic status updates.

Watson can only receive PCMU and GSM. Note that Watson's list of codecs may or may not be a subset of the one offered by Bell, as each party indicates the data types it is willing to receive. Watson will send audio data to port 3456 at c.bell-tel.com, Bell will send to port 5004 at boston.bell-tel.com.

By default, the media session is one RTP session. Watson will receive RTCP packets on port 5005, while Bell will receive them on port 3457.

Since the two sides have agreed on the set of media, Bell confirms the call without enclosing another session description:

C->S: ACK sip:watson@boston.bell-tel.com SIP/2.0  
Via: SIP/2.0/UDP kton.bell-tel.com  
From: A. Bell  
To: T. Watson ;tag=37462311  
Call-ID: 3298420296@kton.bell-tel.com  
CSeq: 1 ACK

## **13 Appendix B**

### **Background to NAT**

The Internet is expanding at an exponential rate. As the amount of information and resources increases, it is becoming a requirement for even the smallest businesses and homes to connect to the Internet. Network Address Translation (NAT) is a method of connecting multiple computers to the Internet or any other IP network using one IP address, which allows home users and small businesses to connect their network to the Internet cheaply and efficiently.

In the beginning NAT was thought of as a temporary solution until a long-term solution to the problem of the depletion addresses was developed. The long-term solution was expected to be a proposal for a new Internet protocol IPv6, with larger addresses.

The situation at the beginning of the decade, with IP address depletion and scaling in routing problems led to some technologies which were introduced to solve these problems. There are three approaches:

### **CIDR (Classless InterDomain Routing)**

CIDR is an addressing scheme for the Internet, which allows for more efficient allocation of IP addresses. The original Internet Protocol defines IP address in four major classes of address structure, Classes A through D. Each of these classes allocates one portion of the 32-bit Internet address format to a network address and the remaining portion to the specific host machines within the network specified by the address. One of the most commonly used classes was class B, which allocates space for up to 65534 host addresses. A company who needed more than 254 host machines but far fewer than the 65534 host addresses possible would be wasting most of the block of addresses allocated. For this reason, the Internet was, until the arrival of CIDR, running out of address space much more quickly than necessary.

With CIDR, a single IP address can be used to designate many unique IP addresses. A CIDR IP address looks like a normal IP address except that it ends with a slash followed by a number, called the *IP prefix*. For example:

172.200.0.0/16

The IP prefix specifies how many addresses are covered by the CIDR address, with lower numbers covering more addresses. An IP prefix of /12, for example, can be used to address 4,096 former Class C addresses.

CIDR addresses reduce the size of routing tables and make more IP addresses available within organizations. It makes it possible to have just one routing entry in a router for a whole block of class C networks and introduces some rules how to build these blocks.

The problem of scaling in routing mainly relates to Internet backbone routing, since the backbone routers have to know all networks on the Internet. The ambition with CIDR was to reduce routing entries in the backbone routers, which started to overflow due to the huge number of entries needed for class C networks. When CIDR was implemented that number decreased significantly, which allowed more time for developing long term solutions, especially IPv6.

There is a problem with CIDR and that is when a customer changes the provider but keeps the IP addresses. The old provider still announces the route to the entire block while the new provider must announce a route to the extra net, which has two routes for that net, the CIDR route and the single route. One possible solution is to use the most specific route and another solution is NAT. The first one has the disadvantage of needing a new entry in a backbone router, which CIDR should have prevented. This can be avoided by using NAT, so that the customer keeps the addresses of the first provider for internal use but uses address translation to translate them into addresses of the new provider when communicating over the Internet.

### **Internal IP addresses**

With the growth of TCP/IP technology even outside the Internet more and more enterprises began reserving IP address space for internal communication. So far there was only one global IP pool out of which all addresses were taken, and everyone needing IP addresses got globally unique addresses. In most cases this was not necessary since the majority of enterprises that suddenly needed IP addresses used them only internally, and even when they connected their enterprises networks to the Internet they did not need unique addresses for all their hosts, since for reasons of security and others e.g. caching web traffic, no direct IP connectivity was allowed between internal enterprise computers and hosts on the Internet. It was therefore just a question of time that special IP addresses out of the global pool were reserved for internal IP networks.

For the internal communication the reserved class A, B or C networks can be used.

Obviously these addresses can't be used on the Internet, since they will not get routed. The advantages are that no reservation has to be made in order to get address space and the disadvantage is in an ever changing environment nobody knows if networks that are independently administrated today and have chosen the same address space out of the reserved pool, will be directly connected in the future.

This may be the case within enterprises, where before the network age many smaller networks existed independently, or it may even concern different companies that have to merge their networks for some reason. NAT could be of help in this case.

### **IP address translation**

CIDR served as a short-term solution for the routing table problem and for the problem of address depletion. To make the situation with IP addresses easier the

address space was reserved for internal use, simultaneously IP addresses were only given away for connecting computers to the Internet.

An additional measure is to reuse IP addresses. The intention was that only a small part of hosts communicated across network boundaries at a time, so only those hosts would need a globally unique IP address. The systems IP address can't be changed each time a connection is established outside the internal network, which led to the introduction of a special device, called NAT. The NAT device assigns a global IP to a connection dynamically, to exchange the local IP numbers in the IP packets with the global IP addresses, since the process should be transparent for both end systems. That means only a relatively small number of global IP addresses is needed and only that amount of hosts can communicate across the borders of your network simultaneously. The disadvantages are that your hosts are not reachable from the outside, that the number of simultaneous connections is limited or that the process might not be completely transparent due to the fact that there are protocols like FTP, that transmit their IP address to the other host.

A special form of this approach to NAT is to have one official address and to only use this address for all communication. To allow more than one host to communicate at a time not only the IP address, but also the TCP port numbers are replaced, using a different port number for each connection. Only the number of ports available for the outgoing connections limits the number of simultaneous connections. This is also called NAPT-PT (Network Address Port Translation and Protocol Translation).

## 14 Appendix C

### SIP URL syntax <sup>1</sup>

SIP-URL	= "sip:" [ userinfo "@" ] hostport url-parameters [ headers ]
Userinfo	= user [ ":" password ]
User	= *( unreserved   escaped   "&"   "="   "+"   "\$"   "," )
Password	= *( unreserved   escaped   "&"   "="   "+"   "\$"   "," )
Hostport	= host [ ":" port ]
Host	= hostname   IPv4address
Hostname	= *( domainlabel "." ) toplabel [ "." ]
Domainlabel	= alphanum   alphanum *( alphanum   "-" ) alphanum
Toplabel	= alpha   alpha *( alphanum   "-" ) alphanum
IPv4address	= 1*digit "." 1*digit "." 1*digit "." 1*digit
Port	= *digit
url-parameters	= *( ";" url-parameter )
url-parameter	= transport-param   user-param   method-param   ttl-param   maddr-param   other-param
transport-param	= "transport=" ( "udp"   "tcp" )
ttl-param	= "ttl=" ttl
ttl	= 1*3DIGIT ; 0 to 255
maddr-param	= "maddr=" host
user-param	= "user=" ( "phone"   "ip" )
method-param	= "method=" Method
tag-param	= "tag=" UUID
UUID	= 1*( hex   "-" )
other-param	= ( token   ( token "=" ( token   quoted-string )))
headers	= "?" header *( "&" header )
header	= hname "=" hvalue
hname	= 1*uric
hvalue	= *uric
uric	= reserved   unreserved   escaped
reserved	= ";"   "/"   "?"   ":"   "@"   "&"   "="   "+"   "\$"   ","
digits	= 1*DIGIT

<sup>1</sup> Taken from RFC 2543 SIP: Session Initiation Protocol [9]

## 15 Appendix D

### Enabling 6to4 in FreeBSD

The following is the file that enables our 6to4 in FreeBSD

```
# This file now contains just the overrides from
/etc/defaults/rc.conf
# please make all changes to this file.

# Enable network daemons for user convenience.
# -- sysinstall generated deltas -- #
inetd_enable="YES"
usbd_enable="YES"
keymap="swedish.cp850"
keymap="swedish.iso"
hostname="venus.europe.nortel.com"
moused_flags=""
moused_enable="YES"
sshd_enable="YES"
ntpdate_flags="Time1.Stupi.SE"
ntpdate_enable="YES"

#enable v4 interface
ifconfig_dc3="inet 10.115.32.3 netmask 255.255.255.0"
ifconfig_xl0="inet 141.251.192.121 netmask 255.255.255.0"

#for enabling 6to4
ipv6_enable="YES"
#ipv6_router_enable="YES"
ipv6_gateway_enable="YES"
ipv6_network_interfaces="lo0 dc2"
ipv6_prefix_dc2="2002:0a73:2003"
stf_interface_ipv4addr="10.115.32.3"
```

## 16 Appendix E

### Configuration files

The following describe how we configured our workstations Mercury, Venus, Terra, Mars and Saturn.

#### Mercury

Relevant configuration of mercury:

---

```
/etc/host.conf:
# $FreeBSD: src/etc/host.conf,v 1.6 1999/08/27 23:23:41 peter Exp $
# First try the /etc/hosts file
hosts
# Now try the nameserver next.
bind
```

---

```
/etc/hosts:
# $FreeBSD: src/etc/hosts,v 1.11.2.1 2000/08/18 18:29:19 ume Exp $
#
# Host Database
#
::1                localhost localhost.perihelion.tb mercury.perihelion.tb
127.0.0.1          localhost localhost.perihelion.tb mercury.perihelion.tb
#
```

---

```
/etc/rc.local
# This file now contains just the overrides from /etc/defaults/rc.conf
# please make all changes to this file.
# Enable network daemons for user convenience.
# -- sysinstall generated deltas -- #
inetd_enable="YES"
usbd_enable="YES"
hostname="mercury.perihelion.tb"
moused_flags=""
moused_enable="YES"
sshd_enable="YES"
ntpdate_flags="Time1.Stupi.SE"
ntpdate_enable="YES"

#for enabling v6 /Lars
ipv6_enable="YES"
ipv6_network_interfaces="lo0 dc2"
```

---

```
/etc/rc.local:
# added by lars on 001121
# start name server
/usr/local/sbin/named -c /etc/namedb/named.conf
```

---

```
/etc/namedb/resolv.conf:
```



```

@      IN      SOA    venus.perihelion.tb.  patrik.nortelnetworks.com (
                                1
                                86400
                                3600
                                604800
                                86400 )

                                IN      NS      venus.perihelion.tb.
::2002:a73:2003:0:280:c8ff:feca:ce8b IN      PTR    venus.perihelion.tb.
-----
/etc/namdb/db/perihelion.tb:
$TTL  3600

@      IN      SOA    mercury.perihelion.tb  patrik.nortelnetworks.com (
                                1      ; serial
                                10800
                                3600
                                604800
                                86400)

                                IN      NS      mercury
localhost      IN      AAAA  ::1
mercury        IN      AAAA  2002:a73:2003::280:c8ff:feca:cebb
venus          IN      AAAA  2002:a73:2003::280:c8ff:feca:ce8b
saturn         IN      AAAA  2002:a73:2003:0:280:c8ff:feca:ce9b

```

### **Venus**

Relevant configuration of venus:

```

-----
/etc/rc.conf
# This file now contains just the overrides from /etc/defaults/rc.conf
# please make all changes to this file.

# Enable network daemons for user convenience.
# -- sysinstall generated deltas -- #
inetd_enable="YES"
usbd_enable="YES"
hostname="venus.europe.nortel.com"
moused_flags=""
moused_enable="YES"
sshd_enable="YES"
ntpdate_flags="Time1.Stupi.SE"
ntpdate_enable="YES"

#enable v4 interface /Lars
ifconfig_dc3="inet 10.115.32.3 netmask 255.255.255.0"

#for enabling 6to4 /Lars

```

```
ipv6_enable="YES"  
#ipv6_router_enable="YES"  
ipv6_gateway_enable="YES"  
ipv6_network_interfaces="lo0 dc2"  
ipv6_prefix_dc2="2002:0a73:2003"  
stf_interface_ipv4addr="10.115.32.3"
```

---

```
/etc/rc.local  
# Should probably start nat-pt but that is done by hand still  
*nothing of interest*
```

---

```
/etc/host.conf  
# $FreeBSD: src/etc/host.conf,v 1.6 1999/08/27 23:23:41 peter Exp $  
# First try the /etc/hosts file  
hosts  
# Now try the nameserver next.  
bind
```

---

```
/etc/hosts  
# $FreeBSD: src/etc/hosts,v 1.11.2.1 2000/08/18 18:29:19 ume Exp $  
#  
# Host Database  
#  
::1          localhost localhost.my.domain myname.my.domain  
127.0.0.1    localhost localhost.my.domain myname.my.domain  
#  
10.115.32.3  venus  venus.perigee.tb
```

---

```
/etc/resolv.conf  
search perihelion.tb perigee.tb  
nameserver 2003::10.115.32.4
```

## **Terra**

Configuration for terra:

---

```
/etc/host.conf  
# $FreeBSD: src/etc/host.conf,v 1.6 1999/08/27 23:23:41 peter Exp $  
# First try the /etc/hosts file  
hosts  
# Now try the nameserver next.  
bind
```

---

```
/etc/hosts  
# $FreeBSD: src/etc/hosts,v 1.11.2.1 2000/08/18 18:29:19 ume Exp $  
#  
# Host Database  
#  
::1          localhost localhost.my.domain myname.my.domain  
127.0.0.1    localhost localhost.my.domain myname.my.domain
```

```
#
#
10.115.32.4 terra.perigee.tb terra
-----
/etc/rc.conf
# This file now contains just the overrides from /etc/defaults/rc.conf
# please make all changes to this file.

# Enable network daemons for user convenience.
# -- sysinstall generated deltas -- #
inetd_enable="YES"
usbd_enable="YES"
hostname="terra.europe.nortel.com"
moused_flags=""
moused_enable="YES"
sshd_enable="YES"
ntpdate_flags="Time1.Stupi.SE"
ntpdate_enable="YES"

#configure dc3 interface /Lars
ifconfig_dc3="inet 10.115.32.4 netmask 255.255.255.0"
-----
/etc/rc.local
*basically nothing of interest*
-----
/etc/resolv.conf
search perigee.tb
nameserver 127.0.0.1
-----
/etc/namedb/named.conf
options {
    directory "/etc/namedb";
};

zone "." {
    type master;
    file "db/named.root";
};

zone "0.0.127.IN-ADDR.ARPA" {
    type master;
    file "db/localhost.rev";
};

zone "tb" {
    type master;
    file "db/tb";
};
```

```
zone "perigee.tb" {
    type master;
    file "db/perigee.tb";
};

zone "32.115.10.in-addr.arpa" {
    type master;
    file "db/32.115.10.in-addr.arpa";
};

//A hack to not write so many files.
zone "115.10.in-addr.arpa" {
    type master;
    file "db/tb";
};

zone "10.in-addr.arpa" {
    type master;
    file "db/tb";
};

zone "in-addr.arpa" {
    type master;
    file "db/tb";
};

zone "arpa" {
    type master;
    file "db/tb";
};

-----
/etc/namedb/db/named.root
@ IN SOA terra.perigee.tb. patrik.nortelnetworks.com. (
    1 ; Serial
    10800 ; Refresh after 3 hours
    3600 ; Retry after 1 hour
    604800 ; Expire after one week
    86400 ) ; Minimum TTL of 1 day

    IN NS terra.perigee.tb.

-----
/etc/namedb/db/localhost.rev

@ IN SOA terra.perigee.tb. patrik.nortelnetworks.com. (
    1 ; Serial
    10800 ; Refresh after 3 hours
    3600 ; Retry after 1 hour
    604800 ; Expire after one week
```

86400 ) ; Minimum TTL of 1 day

```

      IN NS terra.perigee.tb.
1     IN PTR localhost.

```

---

/etc/namedb/db/tb

```

@     IN SOA  terra.perigee.tb.  patrik.nortelnetworks.com. (
      1      ; Serial
      10800 ; Refresh after 3 hours
      3600  ; Retry after 1 hour
      604800 ; Expire after one week
      86400 ) ; Minimum TTL of 1 day

```

```

      IN NS terra.perigee.tb.
perihelion      IN NS mercury.perihelion
mercury.perihelion  IN A  10.115.32.15

```

---

/etc/namedb/db/perigee.tb

```

@     IN SOA  terra.perigee.tb.  patrik.nortelnetworks.com. (
      1      ; Serial
      10800 ; Refresh after 3 hours
      3600  ; Retry after 1 hour
      604800 ; Expire after one week
      86400 ) ; Minimum TTL of 1 day

```

```

      IN NS terra.perigee.tb.

```

```

localhost      IN A  127.0.0.1
terra          IN A  10.115.32.4
venus         IN A  10.115.32.3
mars          IN A  10.115.32.2

```

---

/etc/namedb/db/32.115.10.in-addr.arpa

```

@     IN SOA  terra.perigee.tb.  patrik.nortelnetworks.com. (
      1      ; Serial
      10800 ; Refresh after 3 hours
      3600  ; Retry after 1 hour
      604800 ; Expire after one week
      86400 ) ; Minimum TTL of 1 day

```

```

      IN NS terra.perigee.tb.
2     IN PTR mars.perigee.tb.
3     IN PTR venus.perigee.tb.
4     IN PTR terra.perigee.tb.

```

### **Mars**

Relevant configuration of mars:

Mars is currently down but should in essence be configured as venus.  
Probably like below:

---

---

```
/etc/rc.conf
# This file now contains just the overrides from /etc/defaults/rc.conf
# please make all changes to this file.
```

```
# Enable network daemons for user convenience.
# -- sysinstall generated deltas -- #
inetd_enable="YES"
usbd_enable="YES"
hostname="venus.europe.nortel.com"
moused_flags=""
moused_enable="YES"
sshd_enable="YES"
ntpdate_flags="Time1.Stupi.SE"
ntpdate_enable="YES"
```

```
#enable v4 interface /Lars
ifconfig_dc3="inet 10.115.32.2 netmask 255.255.255.0"
```

```
#for enabling 6to4 /Lars
ipv6_enable="YES"
#ipv6_router_enable="YES"
ipv6_gateway_enable="YES"
ipv6_network_interfaces="lo0 dc2"
ipv6_prefix_dc2="2002:0a73:2002"
stf_interface_ipv4addr="10.115.32.2"
```

---

```
/etc/rc.local
# Should probably start nat-pt but that is done by hand still
*nothing of interest*
```

---

```
/etc/host.conf
# $FreeBSD: src/etc/host.conf,v 1.6 1999/08/27 23:23:41 peter Exp $
# First try the /etc/hosts file
hosts
# Now try the nameserver next.
bind
```

---

```
/etc/hosts
# $FreeBSD: src/etc/hosts,v 1.11.2.1 2000/08/18 18:29:19 ume Exp $
#
# Host Database
#
::1          localhost localhost.my.domain myname.my.domain
127.0.0.1    localhost localhost.my.domain myname.my.domain
#
10.115.32.2  venus  mars.perigee.tb
```

---

```
/etc/resolv.conf
search aphelion.tb perigee.tb
```

```
nameserver 2003::10.115.32.4
```

### **Saturn**

Relevant configuration of saturn:

---

```
/etc/host.conf
# $FreeBSD: src/etc/host.conf,v 1.6 1999/08/27 23:23:41 peter Exp $
# First try the /etc/hosts file
hosts
# Now try the nameserver next.
bind
```

---

```
/etc/hosts
# $FreeBSD: src/etc/hosts,v 1.11.2.1 2000/08/18 18:29:19 ume Exp $
#
# Host Database
#
::1                localhost localhost.my.domain myname.my.domain
127.0.0.1          localhost localhost.my.domain myname.my.domain
#
```

---

```
/etc/rc.conf
# This file now contains just the overrides from /etc/defaults/rc.conf
# please make all changes to this file.
# Enable network daemons for user convenience.
# -- sysinstall generated deltas -- #
inetd_enable="YES"
usbd_enable="YES"
hostname="saturn.europe.nortel.com"
moused_flags=""
moused_enable="YES"
sshd_enable="YES"
ntpdate_flags="Time1.Stupi.SE"
ntpdate_enable="YES"

#for enabling 6to4 /Lars
ipv6_enable="YES"
ipv6_network_interfaces="lo0 dc2"
resolv.conf
#search aphelion.tb when in that domain!!!
search perihelion.tb
nameserver 2003::10.115.32.4
```

### **How to circumvent BIND9 installation problems**

This document describes how to install Bind9 on FreeBSD

Since the normal installation of bind9 did not work properly some manual configuration was needed.

Download bind-9.0.0.tar.gz and bind9.tar, from your favorite place. Unpack bind9.tar in /usr/ports, so that bind9 directory is in /usr/ports/net and move bind-9.0.0.tar.gz to /usr/ports/distfiles/.

Create directory pkg under bind9 and move pkg-\* to that directory. Rename them according to "standard" (see /usr/ports/net/bind8/pkg). Rename distinfo to md5 and move to files directory.

```
In /usr/ports/net/bind9:  
  run "make configure"  
In /usr/ports/net/bind9/work/bind-9.0.0/  
  run "./configure --sysconfdir=/etc --localstatedir=/var"  
In /usr/ports/net/bind9:  
  run "make"  
  run "make install"
```

This will install bind9 into /usr/local/. Make sure your PATH has /usr/local/bin and /usr/local/sbin is before /usr/bin and /usr/sbin.

You're done!

If you intend to run named add the following option in named.conf to make it listen to IPv6 requests:

```
option {  
  listen-on-v6 port 53 { any; };  
}
```