Project Number:	**IST-2001-32603**
Project Title:	**6NET**
CEC Deliverable Number:	**32603/UOS/DS/2.3.1/A1**
Contractual Date of Delivery to the CEC:	30th June 2002
Actual Date of Delivery to the CEC:	31st July 2002
Title of Deliverable:	D2.3.1: IPv4 to IPv6 scoping report for end site networks/universities.
Work package contributing to Deliverable:	WP2
Type of Deliverable*:	R
Deliverable Security Class**:	PU
Editors:	Tim Chown (University of Southampton)
Contributors:	Janos Mohacsi (DANTE), Dimitrios Kalogeras (NTUA), Octavio Medina (ENST-Bretagne), Ivar Skjorseter (Invenia), Stig Venaas (UNINETT), Njål T. Borch (Invenia), Pekka Savola (CSC), Brian Carpenter (IBM), Feico Dillema (Invenia), Patrick Grossetete (Cisco), Chris Edwards (ULanc), Martin Dunmore (ULanc), Jan P. Sorensen (DTU), Christian Schild (WWU/JOIN), Rob Evans (JANET NOSC/UKERNA), Piers O'Hanlon (UCL)

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other

** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Abstract:

This deliverable presents an initial review of site-oriented transition tools for IPv4-IPv6 transition and integration. Theoretical overviews and comments are made for a variety of mechanisms. We also review the current deployment status of these tools within the project participant sites, and seek to identify scenarios to which the mechanisms may (or may not) be most suitably applied. This scooping deliverable will be updated periodically to take the form of a site-level IPv4-IPv6 transition cookbook during the project (at M12, M24 and M36)

Keywords:

IPv6 site transition, IPv6 site migration, IPv6 site transition tools, IPv6 site transition scenarios

Executive Summary

This document is the first deliverable in a series of IPv6 site transition deliverables produced as part of the 6NET project. In this document we:

- Describe the available technologies for IPv4-IPv6 transition
- Give an overview of project participant site transition tool deployment status and plans
- Describe a set of transition scenarios, commenting on the applicability of different mechanisms
- Briefly discuss potential methods for evaluating mechanisms

Following on from this scoping report, the project will produce a site transition cookbook for M12, M24 and M36 of the project. The guide will offer a blend of theory, practical advice and considerations for sites wishing to deploy IPv6 services.

At this stage little deployment experience is included. We expect the first cookbook report to include initial deployment results and sample configurations.

Table of Contents

Table of Figures

1. Introduction

When the original Internet grew from the adoption of IP Version 4 (IPv4), the IP protocol was in relative infancy, used by a few thousand sites, and often in non mission-critical environments. IPv4 is now established to the point that hundreds of millions of users worldwide rely on the smooth operation of the Internet for a wide variety of uses. Introducing a new version of IP, even for the longer term gain IPv6 represents, is not a light undertaking.

In this document we consider IPv6 transition mechanisms that may be deployed at a site, which in the context of 6NET is invariably a college or university, that wishes to run IPv6 services. The principal goal of such mechanisms is to make services that exist on IPv4 today available to new systems that are running IPv6. In some cases hosts and routers may run both protocols together, in others there may be the need for IPv6-only systems to access IPv4-only services, and vice-versa. IPv6 can only be deployed if transition mechanisms are available. There can be no "flag day" where the network operating status changes from IPv4 to IPv6.

There are a number of broad classes of mechanisms. Running both protocols on the same devices, such that IPv4 and IPv6 co-exist natively on the same infrastructure, is referred to as dual-stack operation. Where native IPv6 connectivity does not exist, transporting IPv6 by encapsulating it in IPv4 is referred to as tunnelling. Translation is used to rewrite packets or headers at the network or transport layer, e.g. where NAT exists today for mapping between public and private IPv4 addresses, NAT-PT serves a similar purpose but with protocol translation added. Application Layer Gateways (ALGs) provide proxy services for applications where the proxy understands both protocols, e.g. an IPv4 web browser can ccess an IPv6 web site if the request is passed via a dual stack web proxy.

The IETF has adopted a wide variety of mechanisms in its work within the ngtrans group [ngtrans] to date. Indeed, such is the variety, and the complexity of the issues, that a hold has been called on new proposals until the requirements for transition, the scenarios, and the applicability of mechanisms are all better understood. Such discussion was begun at the Minneapolis IETF meeting in March 2002, and continued in Yokohama in July 2002. The results from and work undertaken in 6NET will be fed into the IETF ngtrans WG. Indeed, some 6NET partners are already very active within ngtrans, e.g. ENST with the Dual Stack Transition Mechanism (DSTM) tools.

In Section 2 we review existing methods, discussing some of the topical issues at hand. We include some discussion of issues not directly related to the tools themselves. In Section 3 we summarise the current deployment status of the tools within the Activity participants, listing plans to date for future deployments, and some pointers to existing implementations. In Section 4 we discuss three deployment scenarios, and the tools that may be applicable to those scenarios. In Section 5 we briefly overview methods by which the tools can be evaluated. Finally, we give a brief report from the ngtrans WG meeting in Japan in July 2002 in Section 6.

2. Review of site transition mechanisms

This section describes some of the possible mechanisms that could be used in the transition from IPv4 to IPv6. Most of the mechanism presented here makes it possible for an IPv4 node to communicate with an IPv6 node and vice versa but a few of them only makes it possible for nodes using the same IP protocol to communicate.

There are some existing documents and works describing transition tools. There is an ongoing ngtrans WG draft of transition methods [Nordmark02], which is an update to an existing, but aging, introductory RFC [RFC1933]. There is also a similarly aging RFC on routing for IPv6 [RFC2185]. Of more current work, there is an ongoing Internet Draft overviewing all trannsition techniques [Overview]. There have also been earlier reports published by particpants in the 6NET project, e.g. the UK Bermuda 2 project [Bermuda01] and the GÉANT IPv6 trials [GTPv6].

To a large extent, the work to date has been against a rapidly moving target of transition techniques. There are already 14 RFCs in the ngtrans WG, and 18 further Internet Drafts. This is one reason why the IETF currently has a hold on advancing new techniques, while the applicability of existing techniques is better understood, through requirements analysis. The work of this project can of course be fed in to that process.

When deploying IPv6 at a site one would ideally have an IPv6 router on every physical link where there is a host that is to use IPv6. This could be hard to accomplish in practice though. A large site would perhaps enable IPv6 on just a few links, and expand over time as needed. The cost of upgrading or buying new routers for IPv6 might be too high. One solution to this could perhaps be to have non-IPv6 routers bridge IPv6 packets, but there are better solutions based on tunneling. For tunneling to scale one should ideally avoid manual tunnel configuration (except perhaps at the early phases of site deployment where the number of participating devices is low), and the data flow should follow the IPv4 infrastructure as much as possible. If there are two IPv6 hosts on the same link, they should be able to communicate directly without involving any routers.

2.1. Dual-stack

The first mechanism we present is called "dual stack" [RFC2893]. Using this method we equip a host or router with a double set of protocol stacks in the operating system. IPv4 is handled by one stack and IPv6 by another stack. Each such node, called an “IPv4/IPv6 node”, is configured with both IPv4 address(es) and IPv6 address(es) and can therefore both send and receive datagrams belonging to both protocols. This mechanism uses the protocol version indicator in the physical frame to decide if the IPv4 stack or the IPv6 stack will handle an incoming datagram. For outgoing datagrams the decision of protocol stack is made by the sending application. In other words; each packet, outgoing or incoming, is automatically multiplexed depending on the IP protocol used.

Note there is some distinction between dual-stack and hybrid stack, as it's quite common that the IPv4 and IPv6 stacks can share large pieces of code. So in practice we have single stacks that can talk both protocols, rather than two completely separate stacks, but the approach is loosely referred to as “dual stack”, so we continue to use that terminology here.

2.1.1. Dual-stack hosts and routers

A router that implements dual-stack can route packets belonging to both protocols. It can therefore be connected to native IPv4 networks, native IPv6 networks or networks using both protocols, so called heterogeneous networks. What a dual-stack node cannot do is translate packets

from one protocol type to another. This means that only nodes using the same protocols can communicate.

2.1.2. DSTM (and extensions)

The Dual Stack Transition Mechanism (DSTM) is used where the network infrastructure supports only IPv6, but the nodes on the network may be dual stack, and "legacy" IPv4 applications may wish to communicate to other IPv4 services outside the local IPv6-only network.

2.1.2.1 Introduction

As IPv6 is deployed on the Internet, a number of mechanisms are required to assure proper communication between hosts and networks that use different versions of the IP protocol. Some of these mechanisms are intended for early phases of transition, where only a few IPv6 hosts are introduced in a mainly IPv4 domain. Some others, like DSTM [DSTM], focus on an intermediate stage, where all nodes in a network are IPv6 capable and IPv4 is required only by old applications or to communicate with external hosts over the IPv4 Internet.

Supporting native IPv4 and IPv6 transport in a network domain is a complex task. As a logical step in the transition process, there will be a time where IPv4 support will be turned off inside that domain. From this point on, all communications will take place in IPv6 only. However, it is clear that, during early phases of IPv6 deployment, a number of hosts within the domain will still require IPv4 connectivity.

One solution to this problem is protocol translation [RFC2766]. Whenever a host needs to communicate with an IPv4-only device, native IPv6 packets are sent to a translation box. This equipment transforms IPv6 packets into IPv4 packets and forwards them to the final destination. The reverse action is performed whenever IPv4 packets are intended to a host inside the IPv6-only cloud. This approach presents several drawbacks. Network services not working because of erroneous translation or the impossibility to offer end-to-end layer 3 security are some examples. NAT-PT is discussed in more detail in a later section.

Another solution, the one used by DSTM, is based on the use of tunnels. IPv4 traffic is tunnelled over the IPv6-only domain until it reaches an IPv6/IPv4 gateway. This equipment is in charge of packet encapsulation/decapsulation and forwarding between the IPv6-only and IPv4 domains. The solution proposed by DSTM is transparent to any type of IPv4 application and allows the use of layer 3 security. However, with this scheme, one IPv4 address is required for every host wishing to connect to the IPv4 Internet. DSTM eases this problem by allocating addresses only for the duration of the communication; allowing several hosts to share the same address on a large time scale.

2.1.2.2 Description of the mechanism

The Dual Stack Transition Mechanism (DSTM) is composed of three elements: 1) An Address Server, 2) a DSTM Gateway or TEP (Tunnel End Point) and 3) a Dual-IP node (called a "DSTM node") wishing to communicate using IPv4. Figure 1 presents the interaction between these three elements.

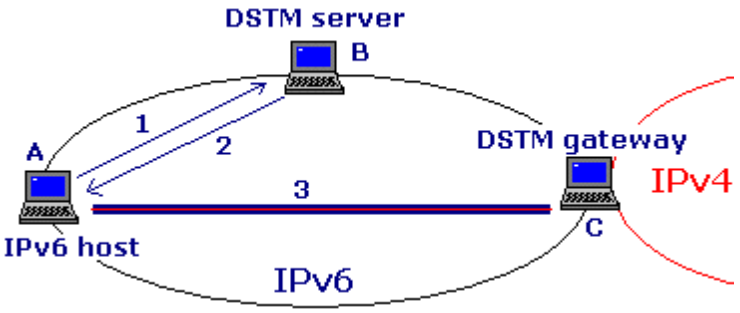

Figure 1: DSTM Architecture

As long as communications can take place in native IPv6, no IPv4 address is required by the DSTM node. The host can detect the need for an IPv4 address by various methods, e.g. a query to the DNS resulting in an IPv4 destination address or an application opening an IPv4 socket.

When the first IPv4 packet needs to be sent, the DSTM client asks the server for an address (1). A number of protocols (DHCPv6 [DHCPv6], TSP [TSP], RPC) can be used for this task. Use of native IPv6 transport is the only restriction for the request.

Following an address request, the DSTM server (who manages an IPv4 address pool) replies to the host (2) with the following information:

- The allocated IPv4 address,
- The period over which the address has been allocated *and*
- IPv4 and IPv6 addresses of the TEP.

This information is used by the node to configure an IPv4 over IPv6 (4over6) tunnel towards the TEP (3). At this point, the DSTM node has IPv4 connectivity and, if it obtained a global address, it will be able to connect to any external host.

The TEP is in charge of packet forwarding between the IPv6-only domain and IPv4 networks. It performs packet encapsulation/decapsulation using an IPv4/IPv6 mapping table; which may be updated "on the fly" or require explicit configuration (by the DSTM server). For successful bi-directional communication, it is required that the reverse IPv4 route to DSTM nodes points to the TEP.

In DSTM, the period of allocation can be configured based on address availability. Nodes are required to ask for allocation renewal every time the allocation time expires. Depending on local policy and node behaviour, the DSTM server may accept or deny to extend the allocation. In normal operation, requests for allocation renewal are periodically sent until the address is no longer needed by the host.

2.1.2.3 Use of DSTM

DSTM is to be used in a network domain where IPv6 routing is enabled and ALL nodes within that domain are able to communicate using IPv6. In this case, IPv4 support can be tuned off: the burden of maintaining an IPv4 addressing plan and supporting IPv4 routing are removed. However, given the huge number of IPv4-only hosts and applications in the Internet, a number of hosts inside IPv6-only domains will still require IPv4 connectivity.

The need for IPv4 connectivity inside a domain can be greatly reduced if ALGs (Application Level Gateways) are properly used. Popular services, such as HTTP or SMTP can take advantage of this possibility. DSTM can be deployed where no other solutions, such as ALGs, can be implemented. DSTM allows Dual IP-layer nodes to obtain an IPv4 address and offers a default route (though a 4over6 tunnel) to an IPv4 gateway. Any IPv4-only application can run over an IPv6-only network if such a scheme is used and, if DSTM is configured to allocate Global IPv4 addresses, hosts inside that domain will be able to communicate with any other host on the Internet.

DSTM may be deployed in several phases. As a first step, IPv4 connectivity may be assured by manually configuring tunnels from Dual-IP nodes to a Tunnel End Point (TEP). In a second phase, when address allocation or tunnel setup protocols become available (DHCPv6, TSP), it would be possible to dynamically assign an IPv4 address to requesting nodes. In this phase, the address may be allocated for the whole lifetime of the requesting node, reducing the complexity of address management. Finally, when IPv4 address availability becomes a problem, DSTM may be configured to allocate addresses only for small periods of time, based on the real needs of requesting hosts.

Since the address allocation process in DSTM is triggered only when IPv4 connectivity is strictly necessary, the size of the IPv4 address pool required by the mechanism should decrease with time (as more hosts and applications become IPv6 aware). However, if the lack of IPv4 address space continues, DSTM may be extended to include the 'ports option' [DSTM-P], allowing simultaneous use of the same address by several hosts, but increasing complexity.

An alternative use of DSTM concerns what has been called "the VPN scenario" [DSTM-V]. It concentrates on the situation where a DSTM node is outside its home domain. Supposing that the node can easily obtain an IPv6 address on the visited network but no IPv4 configuration is possible, the DSTM node can negotiate with its home DSTM server and TEP for IPv4 connectivity. If authentication succeeds and the nomad node obtains an address, the node's IPv4 traffic will be sent to the TEP at its home network using a 4over6 tunnel. Even if the path is not optimal, the node obtains access to private IPv4 resources in its home domain and may obtain global IPv4 connectivity.

2.2. Tunnelling (encapsulation)

The task of encapsulating one protocol packet inside another protocol packet is called tunnelling or encapsulation. This mechanism can be used when two nodes that use the same protocol wants to communicate over a network that uses another network protocol. For example: IPv6 over IPv4 network or IP over Ethernet.

Tunnelling requires two tunnel end-points, each one handling both encapsulation and decapsulation depending on the direction of the packet. The tunnel endpoints must therefore support two protocols; the protocol used to communicate between the endpoints and the encapsulated protocol used by the corresponding nodes. In the general tunnelling case, the tunnel endpoints are dual-stack IPv4/IPv6 nodes.

A tunnel can be configured in four different ways:

1. Router to router, which spans one segment of the end-to-end path
2. Host to router, which spans first segment of the end-to-end path
3. Host to host, which spans the entire end-to-end path

4. Router to host, which spans the last segment of the end-to-end path

[RFC2893] defines two types of tunnels that can be used between dual-stack nodes; configured tunnels and automatic tunnels. Using one of these mechanisms it's possible to connect each native IP island on the Internet independent of the protocol used, IPv4 or IPv6.

The 6BONE [6BONE] is a good example of a tunnelled network using IPv6 over IPv4 (although some parts may be native IPv6).

2.2.1. Configured tunnels

In this case the tunnel endpoint is explicitly configured. The address of the tunnel endpoint is given by the existing configuration information in the node that performs the encapsulation. The encapsulating node must therefore keep information about all the tunnel endpoint addresses. This method can be used in all the four cases mentioned above.

Instead of manually configuring each tunnel endpoint its possible to use executable scripts instead. This "automatic" alternative is called a "Tunnel Broker" (see below) and is presented in [RFC3053]. Tunnel brokers can be implemented as web based applications that make it easy both to configure and administer the different tunnels.

2.2.2. Automatic tunnels

This type of tunnel mechanism uses IPv4-compatible IPv6 addresses on the tunnel endpoints. The address of the recipient node is specified by the packet that is being encapsulated. This method can only be used on router-to-host and host-to-host communication since these are the only ones serving as both tunnel endpoint and recipient. Also, because of the particular addresses used it only works on IPv6 over IPv4 tunnelling.

It is currently quite widely felt that automatic tunnelling should be deprecated. One reason lies in the ad-hoc nature of connectivity that results, lacking structure in the IPv6 domain; solutions such as 6to4 are generally considered preferable.

2.2.3. 6to4

The transition mechanism known as 6-to-4 [RFC3056], is a form of automatic router-to-router tunnelling that uses the IANA-assigned IPv6 TLA prefix 2002::/16 to designate a site that participates in 6-to-4. It allows isolated IPv6 domains to communicate with other IPv6 domains with minimal configuration. An isolated IPv6 domain wishing to communicate with other IPv6 domains will assign itself a prefix of 2002:V4ADDR::/48, where V4ADDR is a global IPv4 address configured on the appropriate interface of the domain's egress router. This prefix has exactly the same format as normal /48 prefixes and thus allows an IPv6 domain to use it like any other valid /48 prefix. In the scenario where 6-to-4 domains wish to communicate with other 6-to-4 domains, no tunnel configuration is needed. Tunnel endpoints are determined by the NLA value (V4ADDR) of the IPv6 destination address contained in the IPv6 packet being transmitted. In this scenario, an arbitrary number of 6-to-4 domains may communicate without the need for any tunnel configuration. Furthermore, the 6-to-4 routers do not need to run an exterior IPv6 routing protocol as IPv4 exterior routing performs the task instead.

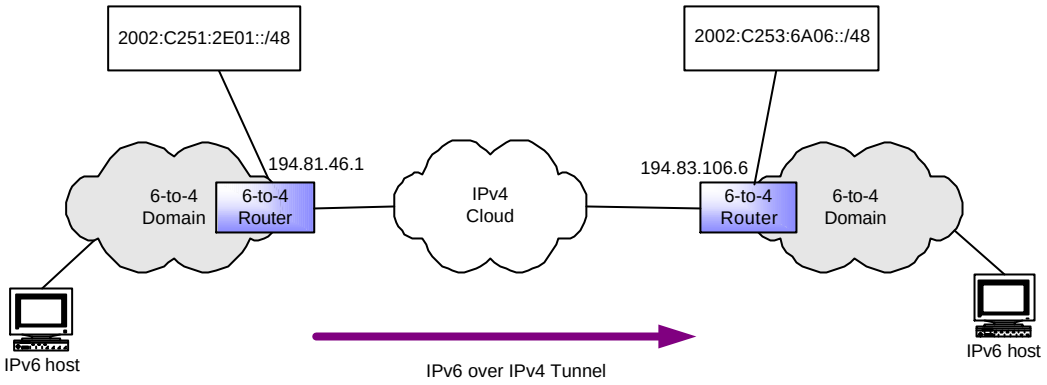

Figure 2: 6to4 service overview

When 6-to-4 domains wish to communicate with IPv6-only domains, the situation is a little more complex. In this case, connectivity between the domains is achieved via a relay router, which is essentially a router that has at least one logical 6-to-4 interface and at least one native IPv6 interface. Unlike the previous scenario, IPv6 exterior routing must be used. The relay router advertises the 6-to-4 2002::/16 prefix into the native IPv6 routing domain. In addition the relay router may advertise native IPv6 routes into its 6-to-4 connection.

The benefit of 6-to-4 is that it simplifies the connection of IPv6 domains that are separated by IPv4-only networks. Minimal configuration is needed and only a single global IPv4 address is sufficient to obtain a /48 IPv6 site prefix. The IPv6 connections inside the 6-to-4 domain can be of any nature e.g. native, tunnelled, 6-over-4 etc. A site can use 6-to-4 tunnelling over a long transitioning period while it is migrating to native IPv6 connections. Since 6-to-4 can be used in conjunction with native IPv6 connections, it can continue to be used until no longer needed (i.e. all external site links have been upgraded to native IPv6).

Unlike 6-over-4, 6-to-4 cannot assume the general availability of wide-area IPv4 multicast, and so must assume only unicast capability from the IPv4 carrier network. One solution to support multicast over 6-to-4 networks is to encapsulate the IPv6 multicast packets within IPv4 unicast packets, and replicating the unicast packets as necessary when multiple branches of the multcast tree occurs [Thaler 02].

2.2.4. 6over4

The general motivation for deployment of 6over4 [RFC2509] is to allow isolated IPv6 hosts, located on a physical link which has no directly connected IPv6 router, to become fully functional IPv6 hosts by using an IPv4 domain that supports IPv4 multicast as their virtual local link. It uses IPv4 multicast as a "virtual Ethernet". This offers hosts the capability to form IPv6 link-local addresses over IPv4 multicast domains. The main advantage of this method is that the end hosts do not require IPv4 compatible addresses and can use the extensive presence of IPv4 without special consideration of the underlying links.

2.2.4.1 Mechanism description

The 6over4 mechanism uses IPv4-addresses as interface identifiers and uses these to form link local addresses. The IPv4-addresses need not be globally unique. By utilizing IPv4 multicast 6over4 allows neighbour discovery, stateless autoconfiguration etc. just like IPv6 over Ethernet. This is very efficient and clients can be automatically configured. If they have 6over4 support and

has no direct IPv6 connectivity, they can use stateless autoconfiguration as usual for their 6over4 interface. The IPv6 Neighbor Discovery multicast packets will be encapsulated in IPv4 multicast packets and should reach any 6over4 routers at the site.

With 6over4 one might get a pretty large link where it is harder to control access to the link than with a physical link. Since it is hard to protect hosts on the same link from each other, the link should be within one management domain. 6over4 uses IPv4 multicast addresses from the organization-local address block, but to make sure it stays within the organization, one must filter packets with such addresses on the boundary routers. One should also filter IP protocol 41 (IPv6 in IPv4) on the boundary and only allow packets for known tunnels.

2.2.4.2 Deployment

Even though 6over4 seems like a good solution, there are very few implementations and it's not being used much. The main problem is perhaps that IPv4 multicast isn't widely deployed. As far as we know IPv4 multicast isn't used much at corporate sites, but there are quite a few academic sites using multicast, that could consider 6over4 if implementations were available.

2.2.5. ISATAP

An alternative to 6over4 is ISATAP (Intra-Site Automatic Tunnel Addressing Protocol). ISATAP also uses the sites IPv4 infrastructure as a virtual link, but it does not use IPv4 multicast, so the link is NBMA (Non-Broadcast Multiple Access).

ISATAP like 6over4 creates an interface identifier based on the interfaces IPv4 address. ISATAP supports both autoconfiguration and manual configuration of addresses, but the IPv4 address of the interface will be embedded as the last 32 bits of the IPv6 addresses. As with 6over4, the IPv4-addresses need not be globally unique.

Usually multicast is used for some neighbour discovery operations like address resolution and router solicitations and advertisements. Since the IPv4 address is always embedded in the IPv6 address, address resolution is trivial. For router solicitations to work, the host must somehow have learned of IPv4 addresses of possible ISATAP routers (through DHCP, DNS, manual configuration etc), and will send solicitations as unicast. The router always sends advertisements as unicast and only as a reply to a host's solicitation. Each ISATAP host will regularly send solicitations to the ISATAP routers it knows of.

ISATAP might not scale quite as well as 6over4 but should work well for most uses. It's not quite clear yet how ISATAP hosts should automatically learn of possible ISATAP routers, but there are a few proposals. There are a few ISATAP implementations, and it might become more used than 6over4 since it doesn't require IPv4 multicast.

With ISATAP one might get a pretty large link where it is harder to control access to the link than with a physical link. Since it is hard to protect hosts on the same link from one another, the link should be within one management domain. To avoid outside use, one should filter IP protocol 41 (IPv6 in IPv4) on the boundary and only allow packets for known tunnels. Spoofing of IPv6 source addresses can only be done by spoofing of IPv4 source addresses, and this can be limited by normal IPv4 source address filtering.

2.2.6. Teredo (aka Shipworm)

The Teredo (previously known as 'Shipworm') transitioning mechanism [TEREDO], is a form of automatic tunnelling intended to provide IPv6 connectivity to IPv4 hosts that are located behind a NAT and therefore do not possess permanent, global-scope IPv4 addresses. As illustrated

in the figure below, the Teredo service employs two entities, a Teredo server and a Teredo relay, in order to provide IPv6 connectivity to Teredo clients located behind a NAT. Unlike other tunnelling mechanisms, Teredo encapsulates IPv6 packets in UDP (instead of directly over IPv4). A well-known UDP port is used by the Teredo entities to transmit and receive IPv6 packets over IPv4.

The Teredo server listens for requests from Teredo clients, responding with an IPv6 address for them to use. The Teredo server forwards the IPv4-encapsulated IPv6 packets sent from Teredo clients to the Teredo relay. The server also forwards IPv6 packets received from the Teredo relay, and destined for a Teredo client, to the appropriate IPv4 address and UDP port of the client. The Teredo relay acts as an IPv6 router and forwards IPv6 packets destined for Teredo clients to the Teredo server and forwards IPv6 packets received from the Teredo server to the IPv6 Internet. The Teredo relay also advertises the reachability of the Teredo service into the IPv6 Internet. It is likely that the Teredo server and relay entities would be co-located within a single router.

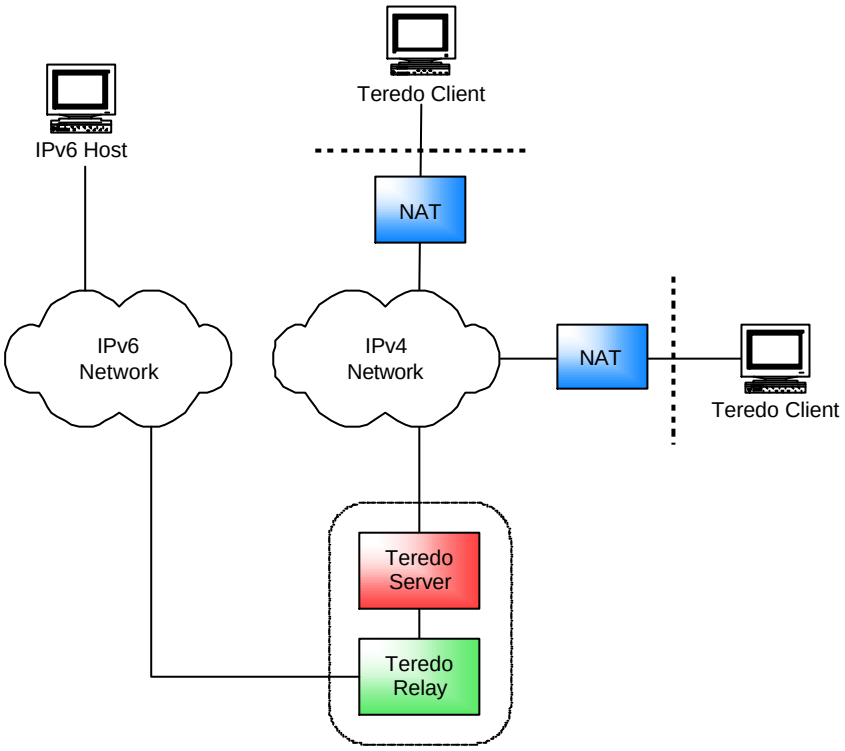

Figure 3: The Teredo Service

Upon boot-up, a Teredo client registers with the Teredo server by composing an IPv6 router solicitation message and encapsulating it with the Teredo IPv4 anycast address and Teredo UDP port (both to be assigned by the IANA). The Teredo server responds with a IPv6 router advertisement (encapsulated in IPv4 and UDP) containing the Teredo IPv6 prefix to be used by the client plus other information required for the client to determine its IPv6 address. This address is not permanent and is only valid for the lifetime of the Teredo service connection (terminated when the client goes offline or the Teredo refresh interval expires). A Teredo IPv6 address consists of a 16-bit prefix (to be assigned by the IANA) that identifies a packet as using the Teredo service. This prefix is followed by a 32 bit field which is the IPv4 address of the Teredo client (as assigned by its NAT), plus a 16-bit field that represents the UDP port being used for Teredo service at the Teredo client. Finally, a random 64-bit field represents the node identifier of the Teredo client.

A Teredo client receives IPv4-encapsulated IPv6 packets on its Teredo service UDP port. These packets are received either from the Teredo server or directly from another Teredo client. When a Teredo client needs to transmit an IPv6 packet it first checks the IPv6 destination address. If the destination is another Teredo client, it can extract the peer IPv4 destination address and peer UDP port from the IPv6 destination address and transmit the IPv6 packet over IPv4 without needing to go via the Teredo server. If the destination is not a Teredo client, the IPv6 packet is sent over IPv4 and UDP to the Teredo server's IPv4 anycast address.

Teredo is intended as a short-term solution to the rather specific scenario of providing IPv6 connectivity to hosts located behind a NAT. As such, it is meant to provide IPv6 access as a 'last resort' to hosts that cannot use any other transition mechanism. It is expected that, over time, NATs will be replaced by IPv6 routers thus rendering the Teredo service redundant. However, this expectation is based on the assumption that NATs are deployed primarily as a tool to overcome IPv4 address shortage. However, it is possible that some organisations will retain their NATs due to other reasons such as security and privacy of their internal address space. Thus, it is possible that the Teredo service may be deployed by sites for much longer than was originally anticipated.

2.2.7. Tunnel brokers

In the case where a user has a dual-stack host in an IPv4-only network, and wishes to gain IPv6 connectivity, one possible solution lies with a tunnel broker service [RFC3053]. The basic philosophy of a tunnel broker is that it allows a user to connect to a web server, (optionally) enter some authentication details, and receive back a short script to run to establish an IPv6-in-IPv4 tunnel to the tunnel broker server.

The operation of a typical tunnel broker service is illustrated in Figure 4. The tunnel broker provider needs to provide a web server (available over IPv4) and a (dual stack) router device capable of accepting set-up commands to create new tunnels to client end-points. It is possible that both functions can be served from one machine.

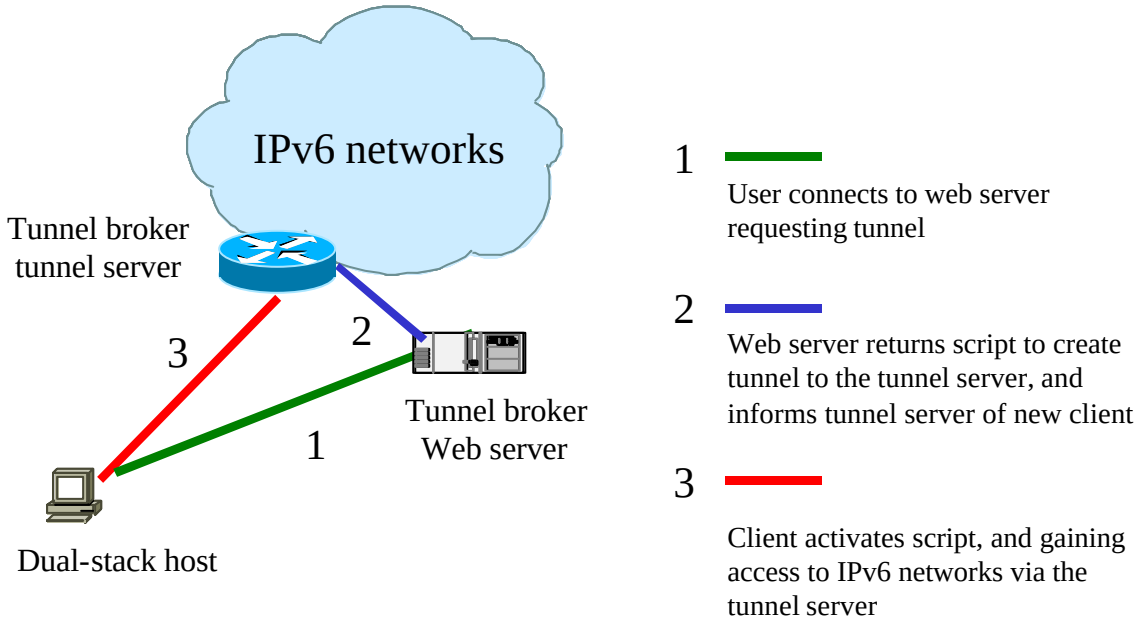

Figure 4: Tunnel Broker components and set-up procedure

Implementation of the tunnel broker can be done in many ways. The service needs to keep track of the tunnels created, and who they belong to. Ideally it should have some authentication to grant access to the service, but in practice early implementations have not required this (the Freenet6 service is perhaps best known, but tunnel brokers have been set up in 6NET project participant networks, e.g. in Norway, the U.K. and Germany).

While the tunnel broker service is generally easy to use for the client, there are some concerns with the deployment of such systems, e.g. in security of access, and in re-allocation of tunnels where clients use dynamic IPv4 addresses (as is typical behind commodity dialup). As with other tunnel methods, any intervening firewall must pass Protocol 41 to/from the tunnel server. Where that is not required, a site administrator may be blissfully unaware of users on their site who use tunnel brokers, thus not creating any site demand for "proper" IPv6 deployment.

A tunnel broker is an important transition aid; it enables easy-to-use IPv6 network access, and we expect to see a number of supported brokers used in the 6NET environment during the project. Brokers may be deployed by sites (universities) or NRENs. If no broker is available to a national participant, remote brokers may be used, but doing so will naturally reduce the efficiency of the tunnelling, since the first IPv6 hop for the client will be in a remote network, even if the target is relatively local.

Tunnel brokers can also serve subnet tunnels, as well as single host tunnels. In such cases the host obtaining the tunnel is in reality a router, and the mechanism for obtaining the tunnel can be more generic (using for example TSP, the tunnel setup protocol [TSP]), and may need specific functions to activate or deactivate the tunnel.

2.3. Translation

If you use either dual-stack or tunnelling mechanisms it's not possible for an IPv6-only node to communicate with an IPv4-only node. This requires translation between IPv4 and IPv6. The translation can occur in a number of different ways, as described in the following subsections.

2.3.1. SIIT, NAT-PT and NAPT-PT

A translator located in the network layer in the protocol stack is called a "header translator". Such mechanisms translate IPv4 datagram headers into IPv6 datagram headers or vice versa. A model that can be used when implementing this mechanism is presented in [RFC2765]: "SIIT – Stateless IP/ICMP Translation Algorithm".

Network Address Translation with Protocol Translation (NAT-PT), defined in [RFC2766], is a service that can be used to translate data sent between IP-heterogeneous nodes. NAT-PT translates a IPv4 datagram into a semantically equivalent IPv6 datagram or vice versa. For this service to work it has to be located in the connection point between the IPv4 network and the IPv6 network.

Just like existing NATs in the IPv4 world translate between (usually) private IPv4 addresses and globally routable IPv4 addresses, the NAT part of NAT-PT translates between a globally routable IPv4 addresses to a IPv6 address or vice versa as well as from a private IPv4 address to a IPv6 address. The PT-part of the NAT-PT handles the interpretation and translation of the semantically equivalent IP header, either from IPv4 to IPv6 or from IPv6 to IPv4. Like NAT, NAT-PT also uses a pool of addresses which it dynamically assigns to the translated datagrams.

Dual-stack and tunnel-based mechanisms do not alter any of the data contained in the datagram. This is true both for IPv4 and IPv6 since the communication is end-to-end using only one

protocol. NAT-PT (and NAPT-PT as described below) on the other hand translate the header of the datagram from IPv6 to IPv4 or vice versa. The result is a new header which is semantically equivalent to the original header but not equal. It's therefore likely that some of the information has been lost during translation. For example that a service, which is only available in one protocol, is lost when converted to another protocol.

[RFC2766] also specifies a service called Network Address Port Translation + Packet Translation (NAPT-PT). This service enables IPvX nodes to communicate transparently using only one IPvY address. NAPT-PT maintains a set of port numbers, which it dynamically assigns to sockets located on the recipient side of the NAPT-PT node.

NAT-PT shares many of the problems as the TRT mechanism, e.g. handling, or failing to handle, IP addresses embedded in application protocol payloads. In particular, there are issues with use of DNS ALGs with NAT-PT [Durand01].

2.3.2. Bump in the Stack (BIS)

The Bump in the Stack (BIS) [RFC2767] translation mechanism is similar to taking the NAT-PT approach with SIIT and moving it to the OS protocol stack within each host. Unlike SIIT however, it assumes an underlying IPv6 infrastructure. Whereas SIIT is a translation interface between IPv6 and IPv4 networks, BIS is a translation interface between IPv4 applications and the underlying IPv6 network (i.e. the network interface driver). The host stack design is based on that of a dual stack host, with the addition of 3 modules, a translator, an extension name resolver, and an address mapper.

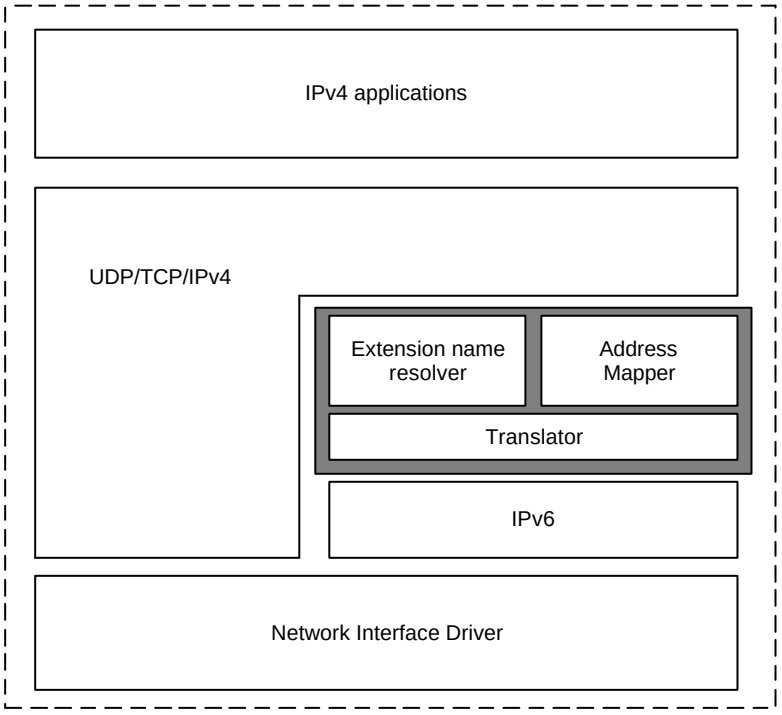

Figure 5: The BIS Protocol Stack

The translator translates outgoing IPv4 headers into IPv6 headers and incoming IPv6 headers into IPv4 headers (if applicable). It uses the header translation algorithm defined in SIIT. The extension name resolver acts as the DNS-ALG in the NAT-PT mechanism. It snoops IPv4

DNS queries and creates another query asking to resolve both 'A' and 'AAAA' records, sending the returned 'A' record back to the requesting IPv4 application. If only 'AAAA' records are returned, the resolver requests the address mapper to assign an IPv4 address corresponding to the IPv6 address. The address mapper maintains a pool of IPv4 addresses and the associations between IPv4 and IPv6 addresses. The address mapper will also assign an address when the translator receives an IPv6 packet from the network for which there is no mapping entry for the source address. Because the IPv4 addresses are never transmitted on the network, they do not have to be globally unique and a private address pool can be used.

The BIS mechanism may be useful during initial stages of IPv4 transition to IPv6 when IPv4 applications remain unmodified within IPv6 domains. However, BIS is limited in its translation capabilities. It allows IPv4 to IPv6 host communication but not vice versa. It does not send or receive any IPv4 packets to/from the network. Thus, even an IPv4 application attempting communication with another IPv4 application using BIS, will fail without additional translation mechanisms somewhere in the communication path. As with NAT-PT and SIIT, BIS will not work for multicast communications and will not work for applications that embed IP addresses in their payloads. An ALG is required for any application that exhibits this behaviour.

2.3.3. Bump in the API (BIA)

The Bump in the API (BIA) [Lee 02] translation mechanism is very similar to that of BIS. However, instead of translating between IPv4 and IPv6 headers, BIA inserts an API translator between the socket API and the TCP/IP modules of the host stack (see Figure 6).

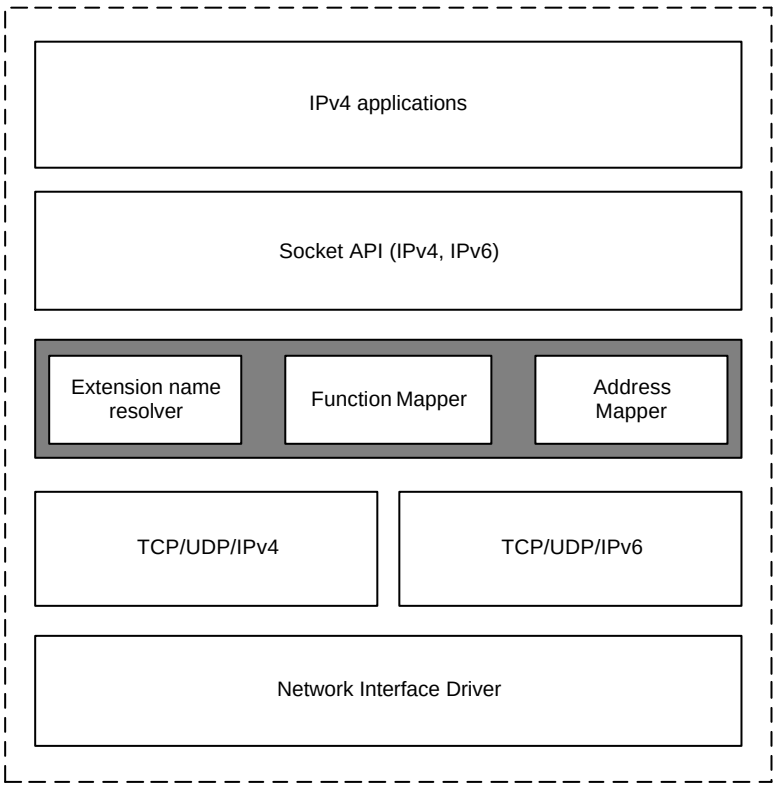

Figure 6: The BIA Protocol Stack

Thus, IPv4 socket API functions are translated into IPv6 socket API functions and vice versa. In this way, IPv4/IPv6 translation can be achieved without the overhead of translating every packet header. Like BIS, BIA is based on the addition of 3 modules: the extension name resolver, the function mapper and the address mapper. Both the extension name resolver and the address mapper modules operate in exactly the same way as the corresponding modules in BIS. The function mapper is the entity that maps IPv4 socket calls to IPv6 socket calls and vice versa. The function mapper does this by intercepting IPv4 socket API function calls and invoking corresponding IPv6 socket API function calls in their place. These IPv6 socket function calls are used to communicate with the peer IPv6 host and are transparent to the IPv4 application that invoked the original IPv4 socket function calls.

The BIA mechanism is intended for systems that have an IPv6 stack but contain applications that have not been upgraded to IPv6. Thus, BIA may be useful in early stages of transition when there are many unmodified IPv4 applications within IPv6 domains. BIA allows IPv4 to IPv6 host communication, but does not specify the reverse case. However, it could be easily extended to cater for IPv6 to IPv4 host communication (this is also applicable to BIS). The advantage BIA has over BIS is that it is independent of the network interface driver and does not introduce the overhead of per-packet header translation. However, BIA exhibits similar limitations to BIS. It will not support multicast communication without some additional functionality in the function mapper module, and it will not work for applications that embed IP addresses in their payloads.

2.4. Transport relay

A translator located in the transport layer is called a transport relay. The relay is located somewhere between the communicating nodes. The nodes could for example be a client and an application server. The relay “picks up" all UDP and TCP sessions that the client tries to set up with the application server. It either relays packets from the client (UDP) or it sets up a connection between the client and relay in such a way that the client thinks it’s communicating with the actual application server (TCP) and between the relay and application server. In this way the relay can hide the actual connection between the two communicating nodes.

2.4.1. TRT

The Transport Relay Translator (TRT) [RFC3142] enables IPv6-only hosts to exchange traffic (TCP or UDP) with IPv4-only hosts. No modification on hosts is required, the TRT system can be very easily installed in existing IPv6 capable networks.

2.4.1.1 Introduction

A transport relay translator which runs on a dual-stack node can use one protocol when communicating with the client and one protocol when communication with the application server. In such a setting the relay is able to translate in the transport layer all data sent between the client and application server. For TCP such a translation includes recalculation of checksum, keeping necessary state about which client socket is connected to witch server socket, and removing this state when the client ends its communication. With UDP the checksum is mandatory when using IPv6 but not when using IPv4.

According to measures mentioned in [RFC2765] most UDP packets on the existing Internet use the checksum, the ones that dodn’t (checksum=0) are generally due to malicious or broken behaviour. A good assumption would be that the relay would have to recalculate the checksum no matter what IP protocol is used.

UDP is not a connection-oriented protocol but when we're dealing with relays it's usually treated like one. Instead of letting the relay set up a mapping between client socket and server socket for each packet sent the relay uses timestamps on this socket mapping state. This eases the work for the relay and the state is automatically removed when the timestamp becomes zero.

An example of this type of relay is presented in [RFC3142]. This RFC describes an IPv6 to IPv4 transport relay translator. An implementation of this technique is called "faith" and was made by the KAME project.

The TRT system can work with most of the common Internet applications: HTTP, SMTP, SSH, etc. The transition mechanism operation is relatively simple:

1. The IPv6-only hostA, in the IPv6 only network, asks the DNS server about the connection endpoint.
2. The specially configured DNS server (similar to that used in NAT-PT) resolves the query, and finds that the end-station has only IPv4 addresses.
3. Assuming that the hostA cannot initiate IPv4 connectivity the DNS translates the IPv4 address (A record) to a special IPv6 address (AAAA or A6 record), consisting of a special prefix plus the embedded IPv4 address.
4. The network is set up such that this prefix is routed to the TRT relay server or anycast address of the TRT relay servers.
5. The TRT relay server receives a packet sent by the IPv6-only host hostA, then extracts the embedded IPv4 address, and initiates a connection/or sends a UDP packet on behalf of the IPv6-only host hostA.
6. The answer from the destination is returned to the TRT relay server, which passes it back to the originator.

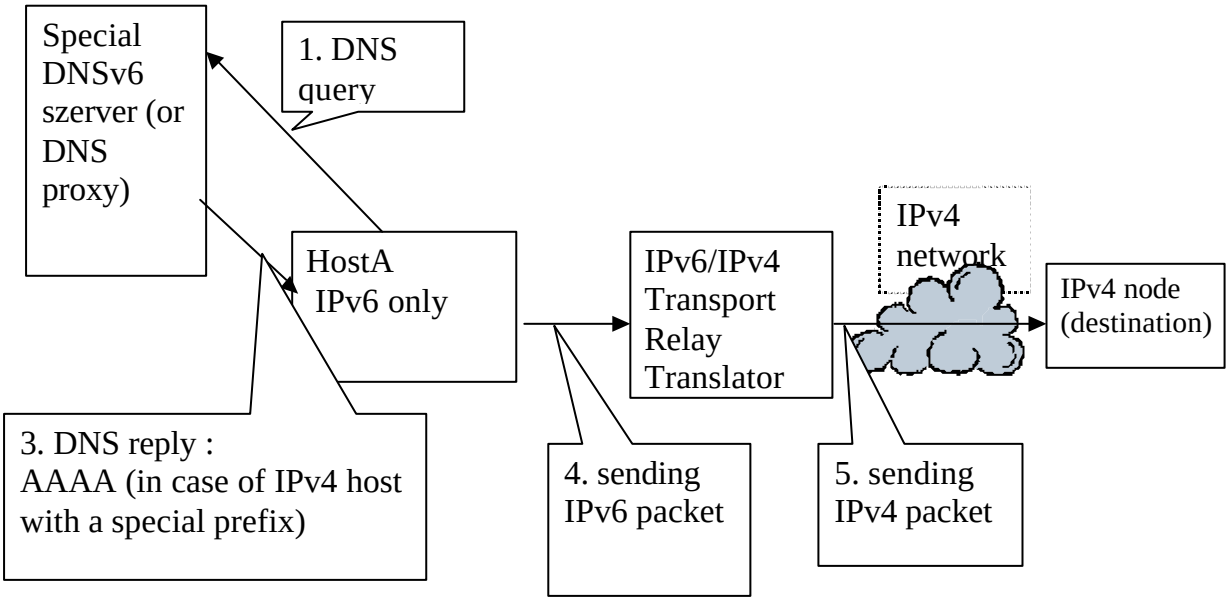

Figure 7: Transport Relay Translator in action

A UDP relay can be implemented in similar manner as a TCP relay. An implementation can recognise a UDP traffic pair like a NAT systems does, by recording address/port pairs into an table and managing table entries with timeouts.

2.4.1.2 Advantages

This technique has the following advantages:

- No problem with fragmentation. If different fragmentation has to be used in the IPv6 and IPv4 parts of the TRT "connections", there is no problem: the Path MTU discovery algorithm or fragmentation mechanism of the TRT relay server can handle the situation.
- No problem with ICMP packets. If any error occurred in any part of the TRT connections the ICMP/ICMPv6 packet is sent back to the TRT relay server, where the error can be handled properly or reported towards to the other end of the "connection"
- Is is not necessary to modify the IPv6 stack of the initiating host, neither is it necessary to modify the application to support relaying.
- It is relatively easy to setup.
- It can be enough to have only one TRT relay server for a whole site. And this router has to have only one global IPv4 address.

2.4.1.3 Disadvantages

The disadvantages of this technique include:

- There can be problems with applications with embedded IP addresses (e.g. FTP, H.323). The TRT relay has to be smart to "look inside" the packets if such an application has to be supported. In this case the TRT relay server becomes a kind of application proxy.
- It supports only unicast TCP/IP traffic as described in [TRT], however it is theoretically not impossible.
- It is more difficult to scale than the stateless translation methods. The TRT relay server has to keep track of all the "TRT" connections to be able properly handle the all the error conditions. The scaling problem can be eased using anycast technology to reach the closest TRT relay server.
- The TRT relay server can generate a major security problem, since it can be used as an intermediate hop to reach IPv4 servers. The served community of a TRT relay server has to be carefully controlled by packet filtering or access control lists. To reduce the problem site local addresses could be used for accepting incoming IPv6 packets.
- It requires a specially configured DNS server to run.
- It requires at least one TRT relay server to be operated per site.
- Due to the nature of the TCP/UDP relaying service, it is not recommended to use TRT for protocols that use authentication based on source IP address (e.g., rsh/rlogin).
- IPsec cannot be used across a TRT relay.

2.4.1.4 Existing Implementations of TRT

Currently only a few known TRT implementation exist, in particular FAITH [FAITH] and pTRTd [pTRTd].

FAITH is an IPv6-to-IPv4 TCP relay, so only subset of TRT specification is implemented. It performs TCP relay just as some of firewall-oriented gateway does, but between IPv6 and IPv4 with address translation. TCP connections initiated from a IPv6 node relayed towards the IPv4 node. FAITH cannot relayconnections for the opposite direction.

To perform relays, FAITH daemon needs to be executed on a dual stack router between local IPv6 site and outside IPv4 network. The daemon is invoked per each TCP services (TCP port number). [FAITH_N]

The Portable Transport Relay Translator Daemon (pTRTd) provide a method for IPv6 hosts to communicate with IPv4 hosts as specified by RFC 3142, similar to the FAITH package implemented by the KAME project. However, unlike Faith, it doesn't depend on special support in the kernel IPv6 stack, and thus should be fairly easy to port to most Unix-like operating systems. Currently only working on Linux system with tun device configured.

2.4.2. SOCKS

SOCKS [RFC1928] is another example of a transport relay but its usually referred to as "proxy protocol for client/server environments".

A SOCKS proxy works in a similar fashion as a traditional transport relay, but there are minor differences which we will now describe.

When a client wants to connect to an application server it first sets up a connection to a well known, preconfigured proxy server using a special proxy protocol. The client informs the proxy about the IP address and port number of the application server it wants to communicate with. The proxy server is now responsible to set up a connection to the application server. As soon as this connection is up and running the proxy relays packet between the client and application server hiding the actual connection.

SOCKS include two primary components: a SOCKS server and a SOCKS client library. The server component is located in the application layer while the client component is located between the client application and the transport layer.

Before a application client can use SOCKS it have to be modified ("socksified"). This can be done in two different ways: If we're in possession of the source code we can compile it together with the SOCKS client library using a set of pre-processor directives. If we do not have the source code, but the operating system supports dynamic linking of shared libraries we can change some environment variables in the operating system so that the client uses SOCKS instead of default network libraries.

[RFC3089] presents a SOCKS-based IPv6/IPv4 gateway mechanism that supports both IPv6 to IPv4 communication and IPv4 to IPv6 communication. This RFC also contains a link to two different implementation of the mechanism described; one from NEC and one from the KAME-project.

2.5. Application Layer Gateways

Application Layer Gateways (ALGs) have much in common with transport relays. In most cases the application knows about the existence of the ALG and is configured to use it. However, in the classic NAT+ALG sense, the application may use the ALG, or proxy, transparently. Whether the end user is aware of the proxy is another issue; the proxy may be discovered by the application without user intervention.

ALG is used by application protocols that make transactional requests towards an application server. Instead of sending the request to the server the client sends it to the ALG. The ALG than sends a request to the server on behalf of the client. The ALG is also responsible for sending all received data from the server back to the client.

An ALG supports only one application protocol. An organization that wants several application protocols to be handled by ALGs must set up one ALG for each application protocol. If the ALG is set up on a dual-stack host it can handle IPv6/IPv4 translation for that particular application protocol it's configured to handle since each "side" of the ALG is treated as having independent connections.

An ALG can also support caching. Since it only handles one protocol the ALG will most likely receive a great number of similar requests. Each request the ALG receives will be checked against all previous request-response sessions. If the ALG finds a match it will send the cached response back to the client. If it was a cache miss the ALG will act just like it didn't have a cache and contact the application server. Usually the use of caching reduces the client's response time, though factors such as server load also come into play.

2.5.1. DNS

The DNS system has an important role when it comes to mechanisms including transport relay translators and NAT-PT: it hides the fact that IPv4-only services requested by IPv6-only nodes are not directly accessible to such nodes but must be routed to and handled by, for example, a TRT. The addresses used in DNS for such services are IPv4-mapped IPv6 addresses where the prefix takes care of the routing to the TRT while the IPv4 address belongs to the node offering the needed service.

Since most services on the Internet are still only accessible via IPv4 we must have a way of offering these services to the IPv6-only nodes with the help of DNS. It's expected that the transition from IPv4 to IPv6 will start with a growing number of IPv6 islands in the sea of IPv4. Most likely there would be a 6-to-4 translator and a DNS translator in many, maybe all of these islands. The advantage of such a solution is that the routing to the IPv4 world would happen locally and the maintenance of DNS would also happen locally.

To have one, or just a few, 6-to-4 translators handling access for all IPv6 nodes to the IPv4 world for the entire Internet would not be a wise decision. Such a solution would at least make a long distance route to reach the 6-to-4 translator, and the DNS would not be managed locally. DNS ALG's can be implemented in two ways, either as an ALG that handles DNS resolver requests from the IPv6 world to addresses in the IPv4 world or as an ALG that converts the data of a zone transfer between a DNS server located in the IPv6 world and a server located in the IPv4 world.

The first solution is to implement an ALG which "picks up" resolver requests from IPv6-only nodes. On behalf of the resolver the ALG sends a request for an AAAA or A6 record of the resource to a DNS server. If the DNS server replies with the correct record it is sent back to the resolver. If the DNS server reply fails, the ALG tries a new request for an A record of the same resource. Most likely this will succeed.

Before sending the reply back to the resolver, the ALG modifies the IPv4 address to an address in the form of an IPv4-mapped IPv6 address and changes the record type to A6 or AAAA (note that A6 is moving to Experimental status, so AAAA will dominate in early IPv6 deployments). This kind of mechanism can either be located on each IPv6-only node which forwards each resolver request to a dual stack DNS server located on the IPv6/IPv4 boundary, or

alternatively we can locate the ALG mechanism on the DNS server and let it handle all resolver requests for the entire IPv6 local network.

The latter solution, to convert all IPv4-specific DNS resources to IPv6-specific DNS resources is not used much in the real world. Not yet at least. This is because the requesting DNS server is configured to handle just specific DNS zones; for example the one most used by the organization. It can therefore not handle requests about the entire DNS tree as normal DNS servers can. Actually it can only handle IPv4-specific data for IPv4 resources (not of any use in the IPv6 world), IPv6-specific data for IPv6 resources and IPv6-specific data for a set of converted IPv4 resources. All IPv6-specific requests to non-converted DNS data would fail. The advantage of this solution is that the IPv6 resolvers requesting data from a converted zone get the correct data right away. The response time for the resolver should therefore be "normal" since no conversion is needed.

Using this ALG mechanism each IPv6-only DNS server is configured to get its zone data via the DNS ALG. To handle name-to-address translation nothing special is done in the DNS server but the DNS ALG converts all IPv4 specific data in the zone reply to IPv6-specific using the 6-to-4 translators IPv4-mapped IPv6 address. To handle address-to-name translation the requesting DNS server must also be configured to request a zone by using the reversed version of the IPv4-mapped IPv6 address. When the DNS ALG receives the request it "picks out" the IPv4 part and sends a request for that zone to a DNS server. In the same fashion as with name-to-address translation, each resource record is checked for IPv4 specific data and converted if necessary.

2.5.2. Web ALGs or proxies

We will look at how a web browser can communicate with a web server where there is no direct reachability using the same IP protocol. We will look at two scenarios. The first is how ALGs can be used to make clients reach a large number of servers, the second is how ALGs can be used to make a specific server available to a large number of clients. In both scenarios we assume that the ALG supports two IP protocols, one for communication with the browser (client) and the other with the server. The ALG may be a simple proxy, but might also do caching etc.

2.5.2.1 Giving web browsers access to web servers through an ALG

We will look at how a site using IPv6-only or with IPv6-only clients can still access IPv4 web servers and vice versa. This is typically servers outside the site, but not necessarily.

This is the cleanest and technically the easiest solution, and most web browsers support it. Whenever the browser makes an HTTP (and often also FTP) request, it contacts the proxy and simply submits an URL to the proxy. The proxy will then do necessary DNS lookups and contact the server through either IPv4 or IPv6. The client may need to use DNS to find the proxy address, but all other DNS lookups are done by the proxy. This also means that no DNS or address magic is necessary. Both client and server are aware that there is a proxy between them and may also know each other's IP addresses.

The idea is that an ALG somehow receives the HTTP (and possibly FTP) traffic, and then acts like a server to the client and vice versa. This can be invisible to both client and server. There are several technical issues though.

First of all, if a client only supports say IPv6 (has only an IPv6 address or has only IPv6 network connectivity) it needs to obtain an IPv6 address for the server before it can start sending packets. If this is an IPv4-only server, we need also intercept DNS so that we can create a magical

IPv6 address for it. This problem is shared with the lower layer translation techniques. The browser cannot follow URLs with IPv4 addresses either.

Another problem is how to know what TCP sessions are HTTP. The typical attempt to solve this is to blindly assume that all TCP traffic initiated to port 80 is HTTP. This may of course not be the case. There might also be HTTP traffic on other ports, but it's not very practical to investigate all TCP sessions.

Finally we need the HTTP traffic to reach the proxy, and it must be able to receive and send data using the IP address that the client specified for the server (the destination address of the packets sent by the client). A TCP session is created between the client and the server. This is very much like transport relays. Unless all traffic to the destination address used is routed through the box that contains the proxy, the traffic must be routed through a box that can redirect HTTP traffic (or usually TCP traffic to port 80) to the proxy. This redirection is then just routing based on port numbers. Unless HTTP 1.1 is used, the IP destination address is needed to know which server to contact. If we're trying to reach an IPv4 server from an IPv6 client, the fabricated address is an IPv6 address that can have an IPv4 address for the server embedded. Unless this is done, the HTTP proxy must learn from the DNS ALG what the real destination is.

2.5.2.2 Making a web server available to clients through an ALG

We will look at how a site with an IPv4-only web server (or with IPv4-only connectivity) can make it available to outside IPv6-only clients. This discussion also applies to scenarios with IPv6-only web server and IPv4-only clients, but for simplicity we will discuss the former. The idea is simple.

You set up a proxy to listen to a say port 80, and also perhaps a specific address, and whenever it receives a new request, it submits a request to one specific web server. If the proxy listens to an IPv6-address, you simply put that in DNS in addition to the web server's IPv4-address, and IPv6- enabled clients will reach the server through the proxy. This is sometimes called a reverse proxy, or if caching is performed an accelerator. If one needs to provide access to multiple IPv4-only web servers, one could use a single proxy that listens to multiple addresses, and use the address to know which server was requested. If HTTP 1.1 is used one could possibly also do this with one address. This is similar to one HTTP server process serving multiple virtual servers.

The techniques described above have been tested by UNINETT using a web cache called Squid, running on Linux, and using Cisco router for redirection, UNINETT has also done interception as described in the latter case, but for IPv4-only.

2.5.3. Other ALGs

Similar techniques can be used with other protocols, a common example being FTP proxies. The general principle, that the ALG or proxy can act as a "dual stack application layer relay", holds, although the case of an FTP ALG is complicated slightly by the use of IP addresses within the payload.

2.6. General transition considerations

In the previous sections we have discussed specific transition tools that can be used as early aids to IPv6 deployment. However, there are also a number of other issues that arise in the transition process; these need consideration, and should be reported on somewhere in the project. There is some overlap in this respect with WP5, which focuses on IPv6 applications and porting.

At this early (scooping) stage we report on some of those issues here. These issues may be tackled later in WP2 or WP5 (thus they may not be expanded upon within the WP2 transition cookbooks).

2.6.1. Some applications only try the first returned address

If you have an application that only uses the first address returned by a DNS query, and you add an AAAA record in the DNS for your server, you might run into problems. Some applications currently only try to connect to the first address returned. This is bad if the host isn't reachable over IPv6 for some reason (which could be due to a network failure, or that a host is IPv6 enabled, but has no IPv6 router on its network).

A more dramatic example might be a commercial mail server that did this. So if your MXes have both AAAA and A records, you might have a problem. One instance was discovered when someone had an IPv6 enabled mail server with no IPv6 connectivity (they didn't even know what IPv6 was). A solution might be to have at least one MX with only an A record. The issue of MX handling for dual-stack hosts is discussed in a current Internet Draft [SMTP02].

2.6.2. BIND issues and mapped addresses

On many platforms one can receive IPv4 packets on IPv6 sockets, so the application can use just one socket for both. But if such an application is used on a platform that doesn't allow this, it will receive IPv6 only traffic.

On, say, standard Linux (as opposed to USAGI Linux), you can't bind and listen to IPv4-any after you've bound and listened to IPv6-any. This is okay unless the application tries to bind to both, and terminate on errors. Because of this OpenSSH only binds to the first address returned by getaddrinfo() on Linux. This was also done for X11 forwarding and was fine until the code also allowed binding to localhost. When that was done it would listen to ::1 but not 127.0.0.1 which is a problem for non-IPv6 X-clients. This is fixed now, but is an example of the problems that can arise. Note with USAGI linux one can open two seperate sockets if the IPV6_ONLY socket option is set.

2.6.3. Name servers and firewalls blocking or not replying to AAAA requests

Blocking of AAAA requests by firewalls will typically mean that a getaddrinfo() call to get addresses will hang until some time-out takes place, and is at best annoying.

Related is the problem of nameservers returning `SERVERR' when queried for a AAAA record. This would happen with some very old bind versions, but will hopefully become less common with time.

2.6.4. A library for dual stack programming

Invenia has created a small, experimental network library which provides something called "Network Descriptors". These descriptors are transport endpoints, using either UDP or TCP, and will transparently do all the necessary network interface work for dual stack environments, as well as all the repetitive code of sockets. This includes trying all addresses, listening to multiple sockets, etc. Hence, an application can create one such descriptor, and support both IP flavours. Both unicast and multicast are supported, making it very simple to create applications that support both. As an example, Inevenia wrote a small relay program that relays a stream (tested with MP3 streams) from unicast or multicast to both unicast and multicast, any IP version or both. The library has been tested on both Linux and NetBSD.

Typical usage for a client:

```
ndesc ND = ndOpen("tcp remoteAddress=www.6net.org remotePort=80");
```

```
ndSend(ND, "GET /", 5, NULL); //No target address given - is connected
ndRecv(ND, buffer, sizeof(buffer), TIMEOUT, NULL); //Not interested in
remote address
```

Typical usage for a server:

```
ndesc ND = ndOpen("tcp localPort=80");
ndesc ND2 = ndAccept(ND, TIMEOUT);
ndRecv(ND2, request, sizeof(request), TIMEOUT, NULL); //Not interested
in remote address
ndSend(ND2, reply, sizeof(reply), NULL);
```

The experimental Network Descriptor library addresses these points:

1: A namespace makes creating network descriptors simple yet leaves the interface easily extendible and powerful. Only the address (if remote) and port number is needed, no special address structures must be filled in.

2: Tasks like testing alternative addresses when connecting and looking up names is often done by cutting and pasting a number of code lines. As applications probably always want to do this, the library automates such tasks. It will also hide all the extra code regarding multicast channels, such as filling in address structures and setting socket options to join the channel.

3: Only "UDP" and "TCP" are options in the Network Descriptor library, the flavour of IP is not visible. Also, hostnames are looked up by the library, further removing the need to know about IP addresses. Creating a webserver is done by opening local port 80 and doing a blocking "Accept" on the descriptor.

4: Accept and receive will take a timeout as a parameter. If this parameter is NULL, the operation will be blocking. The application can thus decide when to block, and when to use timeouts. (A server might do a blocking accept but should have timeouts on reads.) There is no need for select calls, file descriptor sets or macros.

Such a library could be useful, but existing abstracted library solutions should also be investigated, as these may be able to be ported to include IPv6 support. There are many issues to consider, such as multihoming, default address selection, SCTP, IPsec and SSL/TLS. These push the library towards being a formal session layer.

XTI and TLI are possible alternatives. TLI has been around for a long time (1986) and it's covered in the standard literature, like Stevens' "UNIX Network Programming". It used to be a SYSV-ism, but XTI, which is an extension of TLI, is part of the X/Open "Network Services" specification. There are probably implementaions for most UN*X flavours available today.

It is most probable that this porting-related activity will be undertaken in WP5, but initial discussion has occured in here in WP2.

3. Current transition mechanism deployment status

In this section we review the current status of deployment of the classes of transition tools described in the previous section. The aim is to identify which partners are deploying which techniques, and perhaps which methods we will not consider in our experiments.

3.1. Partner status and Future Plans

	MM	Dual stack	DSTM	6to4	6over4	ISATAP	Teredo	Tunnel Broker	SIIT, NAT-PT	BIS, BIA	TCP relays	ALGs
CISCO	2	Y	-	Y	-	Y	-	-	Y	-	-	-
IBM	3	Y	-	Y	-	-	-	-	-	-	-	Y
RENATER (ENST)	2	-	D	-	-	-	-	-	-	-	-	-
ACONET	2	-	-	-	-	-	-	-	-	-	-	-
GRNET	2	D	-	-	-	-	-	-	-	-	-	-
INFN-GARR	3	-	-	-	-	Y	-	-	Y	-	-	-
UCL	2	D	-	-	-	-	-	-	D	-	-	-
UoS	12	D	Y	D	-	Y	Y	D	D	-	Y	D
ULANC	10	D	Y	Y	-	Y	Y	D	Y	-	-	Y
UNINETT	2	D	Y	D	-	D	-	D	-	-	-	D
CSC (FUNET)	4	D	Y	D	-	-	-	-	-	-	D	D
UNI OULU	4	D	-	-	-	Y	-	-	Y	-	Y	Y
INVENIA	6	D	-	D	-	-	-	Y	-	-	Y	Y
WWU (JOIN)	7	D	-	D	-	D	-	D	Y	-	Y	Y
DTU	1	D	-	-	-	-	-	-	-	-	-	D
ULB	0	-	D	-	-	-	-	-	D	-	-	-

Table 1: Transition techniques - current and planned deployments

The currently planned deployments are summarised in Table 1. The key for this Table is:

D = Partner has some deployment already

Y = Partner has probable plans to deploy at some point

- = No plans at present to deploy (within 6NET), but may do in the future

Note that Cisco is a special case amongst the partners in that Cisco is developing router product transition tools that will be deployed by the project partners (and anyone else running IOS with IPv6), e.g. ISATAP has been EFT code since December 2001. NAT-PT and 6to4 are also both already available from Cisco. It is likely that additional methods will also be implemented.

We can see from the Table hat the 6over4 and BIS/BIA techniques will not be covered in the scope of 6NET, or if they are they will not be given high priority. This is in line with the current lack of general implementations (and deployment elsewhere) for either technique. BIS is being moved to Experimental by the IETF ngtrans WG. The 6over4 method suffers from a general lack of multicast IPv4 deployment, although universities are perhaps a little different to commercial deployments, often having multicast support on site.

The Table shows that between the partners there is good coverage for tests with important mechanisms such as DSTM, ISATAP, ALGs, tunnel brokers and NAT-PT. The current lack of commitment to Teredo is in part due to that method's stated aim as a "last resort" mechanism, and in part due to a lack of hardening in the IETF ngtrans WG draft status for Teredo.

As and when new methods are proposed, the project will endeavour to adopt and trial such methods where appropriate and where resources permit.

3.2. Notes on partner deployments

Here we overview some notes from a selection of the partner deployments made to date.

3.2.1. Dual stack

Dual-stack deployment is fairly universal across all partners. This is in part to be expected as most universities in Western Europe are not, yet, short of IPv4 space, and can thus afford to run both protocols together.

3.2.2. DSTM

ENST, within the scope of RENATER in France, is the principal architect of DSTM, contributing to the IETF, and making an implementation available. While deployment is currently lacking in other partner sites, may partners have expressed an interest to experiment with the technique, which is a sign of its perceived value.

3.2.3. 6to4

There is already some deployment of 6to4 within Scandinavian partners, and at Southampton. At Southampton, 6to4 is being used to connect students in shared houses to the IPv6 network at the university. UKERNA is at present launching a 6to4 service, including a 6to4 relay, for the UK community (this is deployed as an NREN support tool as reported in D2.2.1).

Invenia Innovation (II) is using 6to4 to access the IPv6 world. This is because Invenia is, at the time of writing, not connected to the 6NET network ye. The machine running 6to4 is of course a dual-stack machine. At the moment Invenia has both IPv4 and IPv6 routing on the local network

containing dual-stack nodes so all services are available. After connection to 6NET, the initial plan is to probably remove IPv4 routing on the local network and use some kind of TRT together with at least totd (a DNS ALG)

CSC uses mainly Red Hat Linux (RHL kernel, no USAGI) for its IPv6 systems. There are a few FreeBSD systems, acting as IPv6 firewall (using ip6fw), 6to4 relay including zebra routing daemon, TCP relay (faith), and a Multicast router (running pim6sd). There is also one Solaris 8 box. Linux is used for access routers, with Cisco 7200's and 7500's for the main routers. CSC offers a 6to4 relay with anycast for everyone to use, including the commercial Internet, for now. It has worked well to date.

DFN (in conjunction with the JOIN project at WWU) has a 6to4 gateway in the University of Regensburg.

3.2.4. 6over4

No partners are running 6over4, nor currently have any plans to do so.

3.2.5. ISATAP

UNINETT is testing an ISATAP deployment under Linux, having deployed an ISATAP router. JOIN is also deploying ISATAP, so there are now multiple ISATAP routers within 6NET, running both Cisco (router) and Linux (host and router) implementations. The Cisco configuration uses a tunnel command of the form "tunnel mode ipv6ip isatap", in conjunction with router advertisements being turned off (as connectivity is via ISATAP tunnels, where host addresses are formed by the '5efe' ISATAP prefix combined with their IPv4 address, e.g. 2001:638:500:201::5efe:40b0:0101 for a host with IPv4 address 64.176.1.1.

As with any tunnel mechanism, firewalls permitting the tunnel can be run over wide area links, e.g. a host in Germany could use an ISATAP router in Norway. However, that would not make optimal use of the IPv4 routing, so is naturally not recommended.

3.2.6. Teredo

There are no Teredo deployments yet, although some partners are interested, perhaps with IPv6 access for home users with IPv4 NAT over dial-up or ADSL in mind. It should be remembered that Teredo is presented as a "last resort" mechanism.

3.2.7. Tunnel broker

There is a tunnel broker in the University of Leipzig. There is also one being run at Southampton, using OpenSSH over IPv6 to configure the tunnel server and OpenLDAP over IPv6 to store the tunnel allocation schema. Lancaster has also implemented a tunnel broker, running on a Microsoft platform and using Access to hold the tunnel information.

3.2.8. SIIT, NAT-PT

ULB is setting up NAT-PT within its IPv6 testbed, which features two IPv6 Cisco routers, an IPv6 DNS, a web server and a multicasting PC router. The plan is to install NAT-PT on a FreeBSD PC or enable the NAT-PT feature on a Cisco router. MClab has just set up NAT-PT at its site and some cooperative testing between the two sites will be carried out shortly.

Southampton has also been running NAT-PT using the FreeBSD implementation from the KAME code. The gateway has been used successfully to translate outbound IPv6 connections to IPv4, and vice-versa to allow external IPv4 nodes to connect to IPv6 servers (e.g. a web server).

UCL has been running a FreeBSD/KAME based NAT-PT server on its IPv6 network. The gateway has been used successfully to translate outbound IPv6 connections, from IPv6 only machines, to IPv4 servers (e.g. a web server).

3.2.9. BIS, BIA

No partners have plans to deploy either method. However, should ETRI join the project as an International Partner, they may wish to report on BIS.

3.2.10. TCP relays, SOCKS gateway, TRT

CSC has a TCP relay configured to enable IPv6 access to a few IPv4-only services (like NNTP). It is unsuitable for providing WWW access, for that you need an ALG. This is because many sites use absolute links in the web pages, and TCP relays naturally cannot parse them (although relative links should work fine). TCP relays are an acceptable mechanism if there are no addresses embedded in the payload.

3.2.11. ALGs

Invenia is the developer of the totd DNS ALG, which they run on their own site. UNINETT has deployed Web ALGs, e.g. in the form of the Squid web proxy.

DTU is running Apache 2.0.39 with IPv6 on a Linux Mandrake 8.2 server. It operates as a reverse proxy for a background Oracle web server. This will also allow IPv6 clients to access the IPv4 background web server.

UCL has deployed the Transcoding Active Gateway (TAG) which provides application level gatewaying of IPv6 and IPv4 RTP/RTCP sessions, with optional transcoding facilities.

A prototype version of IBM's WebSphere product includes the capability to act as an HTTP ALG (or proxy) between IPv6 clients and IPv4 servers or vice versa. Such edge services will be considered in the Applications work package (WP5), led by IBM.

3.3. Notes on available implementations

Here we list some of the implementations of the transition tools that are currently available for testing. This is not yet an exhaustive list, but one will be built for the future project “transition cookbook” deliverables.

3.3.1. Dual stack

Most operating systems and routers support IPv4 and IPv6 together. The maturity of the IPv6 implementation in some hosts and routers is still a little lacking, while many others are quite advanced.

3.3.2. DSTM

The ENST implementation is available from http://www.ipv6.rennes.enst-bretagne.fr/dstm/.

3.3.3. 6to4

Support is present in many routing platforms, e.g. Cisco IOS 12.2T, and Microsoft has support as described at http://www.research.microsoft.com/msripv6/. There are also USAGI Linux and *BSD versions.

3.3.4. 6over4

Currently we believe only the Microsoft stack includes 6over4 support.

3.3.5. ISATAP

There is an implementation of ISATAP for USAGI Linux at http://v6web.litech.org/isatap/. UNINETT has successfully tested ISATAP on Linux using patches from the USAGI project both as host and router, and also together with ISATAP on Cisco routers, since Cisco IOS also has an EFT image for ISATAP.

3.3.6. Teredo

We are not yet aware of any implementations.

3.3.7. Tunnel broker

There is a tunnel broker implementation available from Tromso (from where the Invenia company was created) at http://www.vermicelli.pasta.cs.uit.no/ipv6/students/vegars/index.html. The website contains source code and documentation in PS and PDF format. The advantage of this tunnel broker is that it uses IPSec. The broker is supposed to run on NetBSD while the clients use Perl scripts to set up the connection (Linux, BSD). Both broker and clients are maintained via a web interface.

Southampton also has a tunnel broker, using OpenSSH and OpenLDAP on FreeBSD, available at present from http://www.6pack.org.

3.3.8. SIIT, NAT-PT

There are a number of NAT-PT implementations, including those in FreeBSD and Cisco IOS. Also of interest is the Ultima package from BT, which is a kind of “double headed NAT-PT” that can enable two IPv6 islands to communicate across an IPv4 cloud. Ultima is described at http://ultima.ipv6.bt.com.

3.3.9. BIS, BIA

An implementation of BIA is available from ETRI at http://www.krv6.net/bia/.

3.3.10. TCP relays, SOCKS gateway, TRT

The KAME faithd is an implementation of a TRT.

3.3.11. ALGs

The Invenia totd DNS application layer gateway implementation can be found at their project web site at http://www.vermicelli.pasta.cs.uit.no/ipv6/DNS.html.

The IPv6 version of Squid is available from http://devel.squid-cache.org/ipv6/. This is a web proxy implementation [note: there was a recent security vulnerability reported in Squid which may not yet be addressed in the IPv6 version].

CSC has used FTP and WWW (proxies from the fwtk package) ALG's, to enable access from IPv4-only systems to IPv6 sites. These work acceptably, but some problems have been observed with FTP extensions with some IPv6 FTP daemons as not all the features were implemented.

4. Scenarios for site transition

In this section we present a selection of scenarios in which transition tools may be deployed. At this stage the scenarios are presented from the end user, or user community, viewpoint, rather than by the class of solution from a management perspective (which is the approach the IETF is apparently taking at present – see Section 6). This choice can be revised for the first edition of the site transition cookbook, if necessary, due in M12 of the project.

Each section features a discussion of candidate solutions, which will be developed and expanded for the first transition cookbook.

4.1. Scenario A: individual or home user

This scenario is applicable to an individual at a university wanting connectivity for a dual-stack host, or a user at home wishing connectivity for a host or perhaps a whole home network. The specific limitations of this scenario is that while it may be possible that the user is using their university dial-up service, it is most likely that the access is required over the commodity Internet.

4.1.1. Introduction and requirements

There are a number of technologies that may be used in this scenario, in which the user may be on a fixed line, or roaming with mobile access, e.g.:

- Dial-up access from PSTN (intermittent, static connectivity)
- ADSL or cable modems (always-on, static connectivity) at home
- Access through a standard or GPRS-enabled mobile phone (mobile connectivity)
- Public WLAN access

The user may or may not have a permanent IPv4 public address, and could also be traversing a NAT. Broadly speaking a number of different types of tunnelling mechanisms may provide the required connectivity.

Where ADSL is used, it is more likely that the user will expect connection of multiple devices on a subnet, and may wish to run inbound services (and we quite probably will see the combination of mobile dial-up/GPRS access to a home ADSL network).

Note that an IETF Internet Draft exists on the subject of unmanaged network transition scope [Huitema02]; this is applicable to small home networks of the type considered here. The Draft discusses implications for devices, applications and services.

4.1.2. Discussion of candidate solutions

The solutions naturally depend on the access mechanism. In the case of ADSL access, it is important to separate bridged and routed ATM encapsulation. In the former case, “only” the DSLAM concentrator needs upgrading to support IPv6. In the latter, the ADSL modem would have to support IPv6.

IPv6 over ADSL can be done through four different encapsulations:

1) ATM RFC 1483 routed
2) ATM RFC 1483 Bridged

3) PPPoA

4) PPPoE

As written, 1), 3) and 4) require the ADSL modem or the PC to support IPv6 but 2) only needs the aggregation router to terminate the ATM VC, then route the traffic. On Cisco IOS, it is called Remote Bridging Encapsulation (RBE) and is available in the latest Cisco IPv6 EFT. Cisco is also working on AAA (the latest EFT provides Cisco VSA for IPv6, i.e. IPv6 prefix, route and ACL), Prefix Delegation (with DHCPv6) and Prefix Pools (again, included in the latest IPv6 EFT).

In the absence of native IPv6 access on an ADSL link, the preferred solution is probably 6to4. This may be run on a router (e.g. a BSD PC using the publicly available Alcatel ADSL drivers), as is the case with trial deployments at Southampton. There are likely to be some commercial products offering 6to4 within an ADSL router available soon at a reasonable price; the 6WIND ADSL product is an example of the latter case, although the initial pricing is quite high.

For a single host with a (preferably static) public IPv4 address the tunnel broker solution would seem easy to use. Recently some tunnel broker services have been extended to offer networks rather than host allocations, e.g. Freenet6. A question arises as to who offers the tunnel broker – the broker should be topologically close (in IPv4) to the client, but if the client is roaming, the nearest broker may not be known. Where broker access is authenticated, the user may have to use the broker provided by their university or NREN.

Another candidate for an ISP (which may be a University dial-up service) wanting to offer IPv6 to customers that are behind ADSL modems or other equipment that is IPv4-only, is to set up ISATAP routers close to the customers. ISATAP tunnels can run over any IPv4 infrastructure, but ISATAP support is currently in relative infancy.

So, in the case of routed ADSL the options with traditional mechanisms are dependent on the IPv4 address allocation:

- Public, static addresses - use a tunnel, tunnel broker, or 6to4
- Public, dynamic addresses – use 6to4, or tunnel (broker) with severe limitations
- Private addresses – will likely need some clever tricks, perhaps a tunnel (broker) or ALG services provided by the local ISP, or alternatively Teredo (implementations of which are apparently non-existent as of yet)

ISATAP could be used with all options but is perhaps most applicable to the third case. Note that with ISATAP, you cannot connect subnet(s) at home with IPv6 via a router (e.g. Linux/BSD box) - every node must use ISATAP. Also, ISATAP used by an ISP for multiple customers may also have security implications as it has been assumed in ISATAP designs that the "customers" are from one single organization. Security issues are an important concern, even in academic environments.

For dial-up access over GSM/GPRS/PSTN a ppp connection would usually be employed. Currently it is unlikely that PPPv6 (rfc2472) would be supported thus any of the previously mentioned techniques would be used, employing some form of tunnelling over an IPv4 connection.

With public WaveLAN (802.11b) the connection is likely to be IPv4 again, though the link layer can support IPv6 native transport. Again any of the previously mentioned techniques may be used, employing some form of tunnelling over an IPv4 connection.

Additionally if the user wishes to hang an IPv6 network off their connection to the Internet then they may also want to deploy mechanisms to provide connectivity to IPv6 only devices as mention in the sections on translators, transport relays, and ALG's.

In summary, the choice of mechanism available to a user seeking dial-up, GPRS or ADSL connectivity will depend on a number of factors. In the absence of native IPv6 connectivity methods (which are only beginning to appear now in Japan and Asia), the easiest method for a user may be a tunnel broker or 6to4, both of which are (relatively) widely available now. Advanced users may be happier with manually configured tunnels, or may wish to try ISATAP (where implementations exist). It is possible to use 6to4 in a NAT environment, using appropriate forwarding methods at the NAT device. There are also integrated ADSL routers with 6to4 functionality emerging. Only where all else fails, and NAT is in use, would Teredo be considered, and even then no public implementations exist at the moment.

4.2. Scenario B: University or NREN wishing to perform initial IPv6 experiments

In this case we consider a small lab or group within a university (or NREN) wishing to gain some early experimental IPv6 experience.

4.2.1. Introduction and requirements

It is likely that the site would have ample static, public IPv4 address space at present to avoid the use of NAT. Typical cases within this scenario class may be:

1. A single user in a research group, with a dual stack host, wanting IPv6 access as part of a project or piece of research they are performing.
2. A small lab behind a single router, running an IPv6 testbed, perhaps including Mobile IPv6 and WLAN access for mobility experiments.
3. A group of users, working on IPv6, but scattered among a number of IPv4 subnets on the current university or department network.

In each case, a production service is not required; the aim of the connectivity is to gain early experimental experience with IPv6 networking.

4.2.2. Discussion of candidate solutions

In the first case, Scenario A may be applicable. For a single host test, one would expect to seek a manually configured tunnel (on the assumption the user running the tests had networking skills), probably to the NREN access router(s). If the user's NREN offers a tunnel broker service, then that would also be a natural solution.

In the second case, one would expect the site to set up a manually configured tunnel to the NREN service, or if the NREN supports 6to4, with a 6to4 relay, the lab could run 6to4 on its access router. From the NREN's viewpoint, 6to4 would involve less administrative overhead, but manual tunnels would have less security concerns, and would probably allow the NREN to more readily keep a feel for its client IPv6 service users.

In the third case, one would gain external connectivity by manual tunnels or 6to4 as for the second case. Internally, 6over4 and ISATAP may be useful if the IPv6 testbed users are scattered all over multiple IPv4 subnets, and there would be a significant amount of internal communication. The justification is that 1) if IPv6 users are not commonplace and are present in many subnets, it

may not be feasible to add an IPv6 router there, and 2) if there is a lot of internal communication, optimising the traffic so it does not go via tunnels to a border router is worth the effort.

The security model of both 6over4 and ISATAP depend heavily on the fact that the site border routers filter out any untrusted ip-protocol-41 (IPv6-in-IPv4) packets, and in the case of 6over4, also sets a multicast scope boundary between the site and the Internet. If these steps are not done, it may become possible to send forged packets to the site.

Both models assume that each 6over4 and ISATAP domain is reasonably trusted. Similar trust model in IPv6 is often applied to a single subnet, but whether it is acceptable for the whole site is something to consider. ISATAP IPv6 addresses embed the IPv4 address in the last 32 bits. This reveals some degree of the internal structure of the network and configured IPv4 addresses of the nodes to possible information gatherers. If the site has a strict security policy, this may be an undesirable feature.

While 6over4 and ISATAP serve similar needs, ISATAP is the better bet for longer term development. However, one should not forget the option of using manually configured tunnels within the site testbed, or carrying IPv6 in dedicated VLANs if site technology permits. While 6over4 requires IPv4 multicast (which is probably present on many university sites), it lacks implementation support. ISATAP is in its relative infancy, but there are already working implementations for Cisco IOS and Linux; naturally support within Microsoft Windows 2000/XP would also be welcome.

If an IPv6-only network is being deployed as the testbed, then mechanisms to access external IPv4 applications would be required; candidate mechanisms for this task would include ALGs, DSTM (assuming IPv4 addresses available to the testbed, and dual-stack hosts), or NAT-PT.

In summary, connectivity would generally assume static, public IPv4 address(es) available. Manual tunnels or 6to4 would be used for external links, with ISATAP or manual tunnels for internal links (where required).

4.3. Scenario C: University or Department deploying supported IPv6 services

In this scenario a campus or department wishes to make a production quality deployment of IPv6, offering a service to a majority of users across (most probably) the bulk of its existing IPv4 infrastructure.

4.3.1. Introduction and requirements

To offer production level IPv6 services, it is desirable to have at least IPv4/IPv6 dual stack available everywhere on the network. Today this is not always possible, as present hardware is often not flexible enough to offer full support of IPv6 similar to the highly optimized IPv4 transport. Nevertheless there are software stacks available for most of the currently present hardware and there are mechanisms to transport IPv6 through IPv4 networks. In addition to dual stack deployment one may wish to introduce IPv6-only networks, to gain full use of the enhanced address space and to get rid of the address restrictions in place today for IPv4.

To deploy IPv6 in a larger scale network, several steps are to be taken. First one has to deploy IPv6 in the core of the network. If the core is switched this may be very easy, as only the campus network core's edge has to be considered. Then one has to think about delivering IPv6 to all kind of attached subnets. These vary from simple switched subnets to wireless LANs or even Dial-In networks. And last (or perhaps first) the local IPv6 network needs external connectivity.

4.3.2. Discussion of candidate solutions

4.3.2.1 External connection

An external connection could be either native or an IPv6-in-IPv4 tunnel. It depends on what the facility is willing to spend for neccessary additional external lines for a native connection and on what the facility's provider is able to offer. Some of the participating NRENs already offer native connectivity to their customers.

To connect to an external peer an IPv6-capable router is needed. There are a lot of vendors who support IPv6 [Imps], the most likely routers in use are from Cisco, Juniper or Hitachi. If one has no spare router available and does not want to jeopardise an existing IPv4 network by adding IPv6 support to present routers, a software router on a PC workstation is also possible. Current solutions for software routers include MRT and Zebra.

4.3.2.2 Campus backbone

There are several ways to transport IPv6 over the backbone. If the core is switched, integration of IPv6 may be very easy, e.g. for ATM or VLAN backbones. In these cases one just has to add an IPv6/IPv4 interface to the emulating LAN. Another easily enhanced backbone is an MPLS-backbone (e.g. Cisco), although this is unlikely to be used in a campus environment.

If the backbone is simply routed, one has to install an IPv6/IPv4 dual stack on every node. This will mostly apply to small cores consisting only of edge devices. Again, this might not be available for every type of hardware. To route IPv6 traffic, today a large set of routing protocols is available for IPv6 on different kinds of hardware: IS-ISv6, OSPFv3, RIPv6, iBGP4+ or (perhaps) IGRP.

There are four general classes of campus infrastructure where IPv6 deployment may be made:

1. Introduce a native IPv6 layer 3 infrastructure (in routers and layer 3 switches). It is expected that enterprise equipment supporting IPv6 (and thus dual-stack with IPv4) will become most favoured when widely available.
2. Deploy a native IPv6 router infrastructure (in parallel with the IPv4 infrastructure) where the IPv6 routers handle routing between VLANs (which in turn may also carry IPv4 traffic). One could thus overlay IPv6 VLANs on IPv4 VLANs, or keep them separate.
3. In the absence of full IPv6 router deployment, tunneled “automatic” connectivity can be achieved using ISATAP (which is already available on platforms such as Cisco IOS).
4. Use manually configured tunnels for an initial deployment. While this might not scale, it has clear and controllable management implications.

There may be areas/subnets inside a network, with very old or strange hardware, where IPv6 routing couldn't be distributed. In addition it may be possible, that the complete site network is not under control of a centralised institution, but that some subfacilities in a site/university handle their network on their own. Such subnetworks might be cut off from the (IPv6-)core in many ways. For such subnets a solution might be to place a host or a small router inside such areas, and configure a tunnel to deliver IPv6 to that subnet. An alternative is ISATAP, but you have to configure ISATAP on every system, and while all OSes have IPv6 support, most of them lack ISATAP at present. One

could also consider 6to4 for such an area, as it offers the opportunity to advertise a prefix for that subnet, but 6to4 would probably be the least favoured method.

4.3.2.3 Subnets

There is a large variety of subnets that need connectivity to IPv6. The most simple and most widespread ones are ethernet framed subnets. IPv6 can be easily delivered to those subnets if they are attached to an IPv6 activated router or if the core network is switched or tagged.

There may be the possibility that such a subnet is not connectable natively to the IPv6 core. In these cases the IPv4 infrastructure must be used to transport IPv6 traffic. Solutions to do so are ISATAP or 6over4 to connect single hosts, and IPv6-in-IPv4 tunnels or 6to4 for a complete subnet. With 6to4 the subnet will use a prefix that differs from the site's prefix, which may be undesirable.

Another kind of subnet is a wireless LAN. The access points to WLANs are often switches which can be easily connected to the IPv6 core. If the access point behaves like a router it has to be IPv6-aware, which is highly unlikely in todays available hardware. IPv6 for wireless LANs is discussed in 6NET WP4.

Facilities often offer Dial-In access to their network. This can be Dial-up, ISDN, ADSL or maybe GPRS. Most of these techniques use PPP somewhere to establish a connection. While it is possible to transport IPv6 over PPP, it is not easy to establish a PPP connection without IPv4, e.g. most stacks can't use IPv6CP as NCP. It is most likely that a connection will be established using a private IPv4 address. After that, IPv6 will be tunneled with these IPs via ISATAP or 6to4 if the dialer gets a static IPv4 address, or via a tunnel broker if the IPv4 address is assigned dynamically.

IPv6 for Dial-In access is discussed in Scenario A, so further details described there apply here too.

A special case is an IPv6-only subnet. These will likely be newly built subnets, so connectivity to the IPv6 core is probably easily established. Nevertheless such subnets need access to the IPv4 world. A large set of transition mechanisms for this case is available, a list can be found in this document. Probably wireless LAN is the first kind of subnet where IPv6-only networking will be introduced.

4.3.2.4 Higher level applications

Apart from the network, there is a large set of services and applications that need to be IPv6-aware and that must be thought of, if a university wants to integrate IPv6. First of all, there is the nameservice and DHCP, which have to be enhanced. Then a facility's NOC will want to manage the 'new' network, and last but not least often higher level services like Mail, FTP or HTTP are offered. Work on these subjects is being done in other 6NET WPs, but as this is vital for the integration of IPv6, it is mentioned here.

4.3.2.5 Other considerations

It is possible that some old equipment exists that cannot be upgraded, or that protocols such as X.25 or Frame Relay are in use. This emphasises the fact that “legacy” IPv4 (and other) components will need to be supported for some time (one would estimate at least 10 years) after the first introduction of IPv6.

5. Considerations for mechanism selection

Here we discuss some methods by which transition mechanisms can be evaluated. In many cases direct comparisons between techniques are not necessarily very meaningful. However, we may find it useful later to have a common set of properties by whch to report on the mechanisms, current or future.

As part of the cookbook development process within 6NET, we shall comment on each existing transition mechanism, some evaluation criteria for which are described below. The first cookbook is due in M12 of the 6NET project (end of December 2002).

5.1. Scalability

Perhaps the most important consideration is how a given mechanism will scale. For example, NAT-PT can handle small numbers of connections quite happily (given the appropriate DNS considerations). But, like IPv4 NAT, as the number of devices initiating connections that require translation grows, the load, in terms of processing requirements and state maintenance, grows. With 10,000 concurrent sessions through one NAT-PT device, we might expect to see some degradation in the service. The question is then whether multiple devices can be used to load balance the connections. Similar questions can be asked of ALGs, of ISATAP, and of other mechanisms, although they may not require state maintenance.

It would be useful if tools existed to test mechanism performance under load, either by simulation or traffic generation. Throughput tools such as ttcp or netperf can measure end to end performance to an extent. However, we may need more specific tools to test mechanisms where factors such as translation, encapsulation and state tables are important, to gauge the direct effect of those factors on the performance.

5.2. Security

The security implications of a given transition mechanism should be presented in the section of that name within the Internet Draft or RFC document. There are some types of attack that may be generic to many mechanisms, such as IP spoofing, IP-in-IP tunnels being allowed through firewalls, or relayed denial of service attacks. It may be useful to survey the security sections of all mechanisms to build a list of identified problem areas, such that a checklist can be applied against future mechanisms.

5.3. Performance

The performance of transition mechanisms is quite closely tied with the issue of scalability. However, performance impact can be direct or indirect. Encapsulation of traffic will have a computation impact, and will also affect packet sizes. Indirect effects include the effect on Ipv4 performance where a device is also acting as an encapsulating endpoint for IPv6 traffic.

5.4. Requirements on the hosts and routers

Some mechanisms may place specific requirements on the host, e.g. specific configuration of the mechanism within the network layer. An example of this would be ISATAP, where a host

has to be explicitly configured to use ISATAP tunnelling, instead of (for example) autoconfiguring an IPv6 address from observed IPv6 Router Advertisements.

Where such configuration is required, the cost of introduction of the mechanism would be a factor in deciding whether to deploy that technique.

5.5. IPv4 and IPv6 address requirements

Different mechanisms place different requirements on available IPv4 (and IPv6) addresses. For example, DSTM would call on a pool of IPv4 addresses for the IPv4-in-IPv6 tunnelling to the edge router and onwards to the destination host (although there is a DSTM variant proposed with port translation as well). The 6to4 mechanism requires only one global IPv4 address.

5.6. Requirements on the application

It may also be the case that applications need to be made aware of the transition mechanism being used. An example is with the use of ALGs, where the application needs to be configured to use the given ALG (although the application could also discover the ALG automatically).

Application porting is being covered in 6NET WP5. It is generally desirable to port code such that the software can run over either IP version.

5.7. Ease of use

Transition tool configuration should be hidden from the end user of the application; if IPv6 is successfully deployed, the end users will be unlikely to realise the change in the underlying IP version. By "ease of use" we are thus referring to the user's perspective.

5.8. Ease of management

The "ease of management" property refers to both the deployment effort required (also covered in sections above) and the effort for ongoing management of the transitioning network. There will also be some indirect implications on network management, e.g. the potential for growing numbers of IP-in-IP tunnels passing through firewalls, and the growth in use of end-to-end IPsec (also through firewalls).

Included here would also be interactions between techniques. For example, ISATAP and 6to4 can be used together for external/internal connectivity; other methods do not integrate so well.

5.9. Other factors

There are most likely other factors that have not been identified here yet. In cases where the end user of the IPv6 is technically minded (which appears to be the case for early trial and testbed deployments), personal preferences for techniques may come to the fore, especially where two techniques have very similar implications and costs.

6. IETF ngtrans WG report, Yokohama, July 2002

Here we include a brief overview of the ngtrans discussions held in Yokohama, and their implications for the planned 6NET activities.

6.1. IETF ngtrans WG report

The chairs introduced the design teams that had been set-up to produce documents that detail the transition scenarios for various types of network and provide a context for the IESG to evaluate documents for the various transition mechanisms. Teams have been established to look at:

- 3GPP networks
- Unmanaged (home) networks
- ISP networks
- Enterprise networks

The enterprise network team has only recently started work and had no documents to present at the meeting, and the 3GPP team does not have a direct bearing on 6NET and so is omitted from the report. Each team is expected to produce two types of document, the first defines the scope and introduces the problems, the second presents the possible solutions.

Christian Huitema is leading the team looking at unmanaged networks and has produced the scoping document[Huitema02]. In the context of the document, unmanaged networks are small home or SOHO networks with no experienced network manager. The approach was to start from the point of view of applications and create four groups of applications (local, client, server and point to point), then for each class examine the requirements in terms of connectivity and addresses, naming and security.

The document describing the solutions is not yet available, but Christian described some of the contents. The "chicken and egg" situation of application porting requiring an upfront investment in stacks, which isn't likely to happen unless the stack writers see a requirement from applications was mentioned, but it was pointed out that many, if not all, newly deployed systems include an IPv6 stack, whether or not it is enabled. The suggestion that what was really needed was a flagship application should also be discussed in the document, and the four classes of applications are looked at in that context, with the conclusion that point to point applications are likely to be the primary driver, with server applications second. Client and server applications both work with few issues in the current IPv4 network architecture.

The first draft of the scenarios document for ISPs has also been produced[Mickles02]. This is still in its infancy, and has been structured to describe the architecture of various types of ISP network, including Core, broadband cable, broadband DSL, narrowband dialup and Ethernet to the home. Only a couple of the sections have content in the current version of the draft, and whilst it has been relatively easy to find volunteers to write information for the core network section, finding people experienced in the other areas has been more challenging.

In conjunction with the design teams, applicability documents are being written for the various transition mechanisms. There were presentations on DSTM[Bound02], ISATAP[Templin02] and the BGP tunneling techniques[Lefaucheur]. Also, work is being done on

describing the interactions between the transition mechanisms. Opinions were sought on whether this should be a section in each transition scenario document, or a document of its own. There was no conclusive feedback.

Alain Durand gave a short presentation on some of the issues that may be encountered whilst IPv4 only and IPv6 only hosts co-exist with dual-stack machines, and two NAT mechanisms entitled NAT64 and NAT46[Durand02] that could help to enable communication. In brief, the scenarios described are IPv4-only servers of one description or another (HTTP, DNS, ...) and IPv6-only clients, or vice versa.

Finally there was some discussion on where the group should be heading; should it retire some of the mechanisms that aren't likely to be used? Discussion ensued, but it was felt the group should not limit the choices available to users and network operators.

6.2. Impact on 6NET activities

The scenario and applicability studies being undertaken by the IETF are in their early stages, having only been initiated at the March 2002 IETF in Minneapolis.

The IETF ngtrans WG has identified four scenario study areas. The SOHO and Enterprise areas tie in closely with the Scenario A and Scenario C definitions above, with 3GPP loosely falling into Scenario A, but being somewhat outside the scope of academic studies. The ISP scenario is in line with the NREN scenario, being studied and reported on in WP2 D2.2.1. It would be possible to align the 6NET work to closely match the IETF approach, however if we wish to contribute to the IETF scenario analysis work then a slightly different viewpoint could be beneficial.

There are some properties of the academic networks of 6NET partners that perhaps differentiate the academic networks from fully commercial ones. The main differentiator is that most campuses are not (yet) short of IPv4 address space, having typically acquired old “Class B” allocations in the past (enough for approximately 65,000 hosts, if 100% utilised). It is rare for new, large universities to appear requiring significant IPv4 address space. However, NRENs are now generally subject to RIPE constraints on additional IPv4 address space, so new colleges, and schools being brought online may be using NAT in some cases. In general though, one can expect early IPv6 deployment to be dual-stack. The first IPv6-only deployments are likely to be led in areas such as wireless LANs.

The likely dual-stack deployments mean that transition techniques such as ISATAP, 6to4, tunnel brokers and ALGs will be more prevalent than those supporting IPv6-only deployments (translation techniques, plus again ALGs). However, we cannot predict the take-up rate of IPv6-only networking at this time, so we should continue to report on all areas in the 6NET WP2 studies.

7. Conclusions

It is very clear that there are many transition tools available to assist in the process of migration towards and integration with IPv6 network services. Such is the variety of techniques that the IETF ngtrans WG has placed a hold on progressing existing draft methods to the IESG, and on adotption of new WG draft items. Part of the goal of the 6NET work is to evaluate and trial existing techniques. The project can gain practical experience of the use of the tools, and feed back those results into the IETF processes.

There are many ways to evaluate and compare transition mechanisms. We have presented a selection of factors that can be used when undertaking such evaluations. In parallel, the identification of (academic use) scenarios and subsequent applicability analysis is important. Similarly, evaluation of factors such as security and scalability are also key to instill confidence in the community to deploy robust and reliable transition aids.

We have identified which partners in the project will work on which tools, at least as far as M12 at which point the first full "transition cookbook" will be delivered. This cookbook will then be updated further at the ends of years two and three of the project (M24 and M36), forming a "living document" to aid transition deployments in academic (and other) environments. The cookbooks will expand on the scenario analysis, presenting results of deployments, with configuration examples.

8. References

[6BONE] The 6BONE Network, http://www.6bone.net

[Bermuda01] "Bermuda 2: IPv6 Deployment Study for UK Academic Networks, Transition Report": http://www.ja.net/development/internet2/bermuda-d2-final.pdf

[Bound02] "Dual Stack Transition Mechanism (DSTM) Overview", J. Bound, Internet Draft, June 2002, draft-ietf-ngtrans-dstm-overview-00.txt

[DHCPv6] "Dynamic Host Configuration Protocol for IPv6", Bound, Carney, Perkins, Droms, DHC Internet Draft; draft-ietf-dhc-dhcpv6-16.txt; October 2001.

[DSTM] "Dual Stack Transition Mechanism (DSTM)", Bound, Toutain, Medina et al. NGTrans Internet Draft, draft-ietf-ngtrans-dstm-08.txt, July 2002.

[DSTM-P] "DSTM Ports Option for DHCPv6", Myung-Ki Shin, Internet Draft, draft-ietf-dhc-dhcpv6-opt-dstm-ports-01.txt, June 2002.

[DSTM-V] "DSTM in a VPN Scenario", Richier, Medina, Toutain, Internet Draft, draft-richier-dstm-vpn-00.txt, February 2002.

[Durand01] "Issues with NAT-PT DNS ALG in RFC2766", A. Durand, Internet Draft, January 2002, draft-durand-natpt-dns-alg-issues-00.txt

[Durand02] "NAT64 - NAT46", A. Durand, Internet Draft, June 2002, draft-durand-ngtrans-nat64-nat46-00.txt

[FAITH] "Translating IPv4 and IPv6 connections", Yoshinobu Inoue and Jun-ichiro itojun Itoh, http://www.kame.net/newsletter/19981001/

[FAITH_N] "Using FAITH TCP relay translator", NetBSD documentation, http://www.netbsd.org/Documentation/network/ipv6/#faith_translate

[GTPv6] "IPv6 Testing for GÉANT", GÉANT IPv6 Working Group (GTPv6), http://www.dante.net/tf-ngn/D9.3.pdf

[Huitema 02] "Unmanaged Networks Transition Scope", Internet Draft, June 2002, draft-ietf-ngtrans-unmanscope-00.txt

[Imps] IPv6 host and router implementations, http://playground.sun.com/pub/ipng/html/ipng-implementations.html

[Lee 02] "Dual Stack Hosts Using 'Bump in the API' (BIA)", S. Lee, M. Shin, Y. Kim, E. Nordmark, A. Durand, Internet Draft, draft-ietf-ngtrans-bia-05.txt, April 2002.

[Lefaucheur] "Operational Environments and Transition Scenarios for "Connecting IPv6 Islands across IPv4 Clouds with BGP"", F. Le Faucheur, Internet Draft, June 2002, draft-lefaucheur-bgp-tunnel-transition-00.txt

[Mickles02] "Transition Scenarios for ISP Networks", C. Mickles, Internet Draft, July 2002, draft-mickles-ngtrans-isp-cases-00.txt

[ngtrans] IETF ngtrans Working Group, http://www.ietf.org/html.charters/ngtrans-charter.html

[Nordmark02] "Transition Mechanisms for IPv6 Hosts and Routers", Internet Draft, July 2002, draft-ietf-ngtrans-mech-v2-00.txt

[Overview] "An overview of the introduction of IPv6 in the Internet", W. Biemolt, A. Durand et al, IETF Internet Draft, February 2002, draft-ietf-ngtrans-introduction-to-ipv6-transition-08.txt

[pTRTd] pTRTd, by Nathan Lutchansky, http://v6web.litech.org/ptrtd/

[RFC1928] "SOCKS Protocol Version 5", M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones, IETF RFC 1928, March 1996

[RFC1933] "Transition Mechanisms for IPv6 Hosts and Routers", R. Gilligan, E. Nordmark, IETF RFC 1933, April 1996.

[RFC2185] "Routing Aspects Of IPv6 Transition", R.Callon, D.Haskin, IETF RFC 2185, September 1997.

[RFC2765] "Stateless IP/ICMP Translation Algorithm (SIIT)", E. Nordmark, IETF RFC 2765, February 2000.

[RFC2766] "Network Address Translation - Protocol Translation (NAT-PT)", G. Tsirtsis, P. Srisuresh, IETF RFC2766, February 2000

[RFC2767] "Dual Stack Hosts Using the 'Bump-in-the-Stack' Technique", K. Tsuchiya, H. Higuchi, Y. Atarashi, IETF RFC 2767, February 2000.

[RFC3053] "IPv6 Tunnel Broker", A. Durand, P. Fasano, I. Guardini, D. Lento, IETF RFC 3053, January 2001.

[RFC3056] "Connection of IPv6 Domains via IPv4 Clouds", B. Carpenter, K. Moore, IETF RFC 3056, February 2001.

[RFC3089] "A SOCKS-based IPv6/IPv4 Gateway Mechanism". H. Kitamura, IETF RFC 3089, April 2001.

[RFC3142] "An IPv6-to-IPv4 Transport Relay Translator". J. Hagino, K. Yamamoto, IETF RFC 3142, June 2001

[SMTP02] "SMTP operational experience in mixed IPv4/IPv6 environements", Motonori Nakamura, Jun-ichiro itojun Hagino, , Internet Draft, June 2002, draft-ietf- ngtrans-ipv6-smtp-requirement-06.txt.

[Templin02] "ISATAP Transition Scenario for Enterprise/Managed Networks", F. Templin, T. Gleeson, Internet Draft, June 2002, draft-ietf-ngtrans-isatap-scenario-00.txt

[TEREDO] "Teredo: Tunnelling IPv6 over UDP Through NATs", C. Huitema, Internet Draft, February 2002, draft-ietf-ngtrans-shipworm-06.txt.

[Thaler 02] "Support for Multicast over 6to4 Networks", D. Thaler, Internet Draft, June 2002, draft-ietf-ngtrans-6to4-multicast-01.txt.

[TSP] "DSTM Tunnel Setup using TSP". Blanchet, Medina, Parent, July 2002, Internet Draft, draft-blanchet-ngtrans-tsp-dstm-profile-01.txt.