32603	Deliverable D 6.2.2 **Operational procedures for secured management with transition mechanisms**	6net

Project Number:	**IST-2001-32603**
Project Title:	**6NET**
CEC Deliverable Number:	**32603/Partner/DS/No./A1**
Contractual Date of Delivery to the CEC:	December 2002
Actual Date of Delivery to the CEC:	28th February 2003
Title of Deliverable:	Operational procedures for secured management with transition mechanisms
Work package contributing to Deliverable:	WP6
Type of Deliverable*:	R - Report
Deliverable Security Class**:	PU - Public
Editors:	Ioannis Kappas
Contributors:	Giuseppe Di Battista, Bernard Tuy, Lorenzo Colitti, Jerome Durand, Rob Evans, Dimitrios Kalogeras, Georgios Koutepas, Maurizio Patrignani, Pekka Savola, Stig Venaas

Abstract:

This document examines the operational security issues behind various methods employed when migrating to native IPv6 network interconnection. The most common methods for gradual transition to IPv6 capability and operation parallel to the IPv4 network are presented in detail along with the security issues they raise. Directives for avoiding security problems are given for each solution.

Table of content

1 Introduction

IPv6 is the new version of the Internet Protocol, promising to overcome the lack of IPv4 addresses. It is hoped that a majority of networks in the future will be designed with this new protocol, and existing networks will be migrated in due course. However, this replacement may not be smooth due to incompatibilities on the addressing side (an IPv6 node cannot directly address an IPv4 node and vice-versa). In the years following the first publication of the draft documents describing the IPv6 specification, people willing to introduce or experiment with the new protocol were often faced with the problem of isolation. Although nodes or networks became IPv6 capable, they could not communicate with other IPv6 nodes or networks in the IPv4 dominated Internet except if there was a native link or path of the same type between them. To overcome this problem tunnelling techniques were engineered that have made it possible to connect remote IPv6 nodes or networks together on top of the current IPv4 infrastructure. These mechanisms have been accepted by the Internet community and seen as the vehicle to gradually migrate to an IPv6 Internet.

Most of the fundamental ideas that are found in IPv4 have also been maintained in IPv6 and as such the two protocols are not very different. The additional protocols (e.g. BGP, DNS, SMTP, HTTP) were also upgraded (or there is an ongoing effort to do so) to operate on IPv6. Thus the experience of all the years of deploying, operating and managing IPv4 networks is applicable to IPv6 as well. However, IPv6 carries additional functionality that requires new management practices to be composed and experimented with. One such practice, that is full of potential but not fully understood yet, is the management of IPv4 to IPv6 transition mechanisms. To this end, this document investigates security aspects of the IPv6 migration mechanisms, something that was not applicable to IPv4 and thus has not been studied before.

A lot of work has been undertaken inside the 6NET project to investigate and report on the existing IPv6 mechanisms. This paper is the product of the development and test of management tools activity of WP6. It looks closer at the potential risks and security issues raised by deploying these mechanisms with the overall aim to create awareness to the people that operationally manage the IPv6 and IPv4 co-existence in a production network.

The next section reviews general issues arising by the use of tunnels; section 3 touches upon tunnel brokers, while sections 4, 5, 6 and 7 present the issues a network operator should be aware of when deploying Dual Stack, 6to4, ISATAP and DSTM mechanisms respectively. Section 8 reports on the Tunnel Setup Protocol (TSP) and suggests cases where it can be useful to deploy it, and finally section 9 lists several methods that can help operators discover IPv6 tunnels transported through their IPv4 networks.

Transition mechanisms also encompass translation techniques, such as NAT-PT [NATPT], ALGs etc that operate by rewriting headers or sometimes the packets at the network, transport or even application layer. However they are not covered in this document because of the limited experience this work package has on these techniques at that time. The second version of this deliverable planned for next year should include the experience gained with the translation techniques.

2   Tunnelling

Tunnelling is a very powerful mechanism for IPv6 deployment and transition. By tunnelling IPv6 inside IPv4 one can send a packet through an IPv4-only network to a dual-stack router or host that can remove the IPv4 header and then forward or receive the IPv6 packet inside as usual. There are however some issues one should be aware of.

?? A site may filter some traffic at its border routers or may be using a firewall. If they do not support IPv6 one may want to get IPv6 connectivity by setting up a tunnel from an internal router at the site, to some router outside the site. The internal router then becomes a border router with respect to IPv6. If filtering is done at the border for IPv4 one will probably also want to do the same for IPv6. This filtering should be done at the tunnel end-point. One could consider employing filtering at the border routers or other routers the tunnel passes through, but this is not recommended. First of all, the filtering can be done much more efficiently at the end-point. Secondly it is unlikely that the routers can do this if they do not support IPv6. A firewall could perhaps be configured to look for specific bit patterns in the payload though. This might make it possible to filter on at least IPv6 source and destination addresses; but again, this approach is not recommended. One might also want traffic statistics showing the amount of IPv6 traffic per prefix, port etc. Once more, since peeking inside tunnels is hard, one should inspect the packets when they are outside the tunnel, perhaps at the end-points.

?? When deploying IPv6 inside a site, using a different border router than the one used for IPv4, it is necessary to be sure that IP protocol 41 is not filtered along the path between the two IPv6 tunnel end-points if direct IPv6 encapsulation is used. In the case that GRE encapsulation is used, it must be checked that IP protocol 47 is not filtered between the tunnel end-points. As filtering is usually done at the site level, then this check can be done directly at the network operation centre (or network administrator) of the site that wants to connect or at both end-point sites if the tunnel is aimed at connecting the sites directly (without third intermediate sites). Note that this situation tends to disappear as more and more backbones offer IPv6 connectivity.

?? There might be IPv6 implementations that do not allow the same management operations for tunnel interfaces as for physical interfaces. We have seen at least one host implementation that did not allow tcpdump on tunnel interfaces. There are probably other examples.

?? Purely hardware based routers will need specialised hardware to be able to encapsulate and decapsulate packets so that they can be used as tunnel end-points. Routers that do some operations in hardware and some in software will probably be able to handle this. One should be aware, though, that the software processing power might be limited, and the CPU used for the processing is probably also used for other tasks.

?? A tunnel may pass through a large number of routers and have a large RTT (Round Trip Time) compared to that of a native hop. This is mainly a problem for routing. One might have a path between two geographically distant sites that none the less consists of a very limited number of tunnel hops (e.g. from one country in Europe to Japan, if both connected sites have peerings with the same organisation in Japan) compared to a short native path through 6NET which may be a few hops longer. Without extra configuration, the tunnel path has fewer hops and would be preferred by distance vector protocols (e.g. BGP). However, there are ways to prefer the native path.

?? The MTU for a tunnel link will be less than the path MTU for the tunnel since an IPv4 header will be added to all packets flowing through the tunnel. In general this might lead to undesired fragmentation effects. For IPv6 the use of path MTU discovery makes this much less of a problem. Some IPv6 implementations instead use the minimal IPv6 MTU which is 1280. This will also avoid the problem in most cases, since the IPv4 path MTU is often at least 1500.

3   Tunnel brokers

Tunnel brokers can be a nice way to offer IPv6 to individuals and perhaps small organisations if there is a large number of potential individuals or organisations. The idea is mainly that those that want connectivity can use a web interface or other means to request and configure a tunnel, doing some form of authentication and supplying their IPv4 address etc. The tunnel at the provider end will be configured immediately and the broker might also supply the user with a script to configure their end. Without a broker, each individual would have to contact the provider independently, and an administrator would have to configure each tunnel. So the broker is a way of avoiding some of the administrative tasks, and one that can also offer instant service. It may also offer ways of automatically enabling and disabling the tunnels or ways of changing the configuration. This might be useful if the user does not operate with static IPv4 address. The use of tunnel brokers does raise some issues, as discussed below.

3.1   Authentication

In general one would like to have some idea of who is using the different tunnels, and also make sure that only the owner of the tunnel can change the configuration. If a broker is run by an organisation with directory type structures installed (e.g. a university for their students) one may be able to use usernames and passwords or other means of authentication that the organisation already utilises for other services. If there is no prior relationship with the user, one typical approach is to have people register with name, e-mail address etc. and then receive through e-mail an auto-generated password. This way there is uncertainty about the real identity of the final users are, but one knows that the e-mail address is confirmed as operating and the person receiving the password is one of the receivers of e-mails to this address.

An alternative approach could be to abide by the RFC3129 [KIN]. In this context we accept a Kerberised Internet Negotiation of Keys. Clients authenticate to a centralized server -- the Key Distribution Centre --, which in turn issues tickets that servers can decrypt thus proving that the client is who it claims to be. One of the elements of a Kerberos ticket is a session key which is generated by the KDC which may be used by the client and server to share a secret. Kerberos also allows for both symmetric key authentication, as well as certificate based public key authentication (PKinit).

3.2   Enabling/disabling tunnels

A user might be able to enable and disable the tunnel by using a web interface or perhaps a script using HTTP or other protocol to communicate with the broker. If the user is always connected with a fixed IP address, all should be well. As long as the user leaves the tunnel enabled, it should forward packets. However if a user uses a connection method that does not guarantee a static IPv4 address (e.g. dial-up) and somehow gets disconnected, it would be prudent to take the tunnel down automatically to prevent unnecessary use of resources. More seriously, this would assure that the session is not (intentionally or unintentionally) "hijacked" by someone else who connects with the same address, thus receiving traffic intended for the original user. One possibility would be to automatically take down the tunnel if no packets are received from the user for some time, especially if packets are flowing in the other direction. This could be combined with some utility at the user end that sends some sort of keep-alive messages. The type of packets does not really matter as long as something is sent. One could also have some TCP session between user and broker for controlling the tunnel, and have keep-alive messages sent over the TCP session. If no data arrives or

the session is reset the tunnel can be taken down. The downside to all this is that a solution is harder to deploy if it would require some specialised software for the user operating systems.

3.3   Guarding against misuse and statistics

When running a tunnel broker, one should also apply filtering to some IPv6 packets. The broker (or more correctly, the router that is the end-point for the tunnels the broker manages), should drop any packets received on a tunnel interface, that have a source address not belonging to the remote side (no route for that address in that interface). And it should also not send packets into a tunnel that has source address belonging to the remote side. The first kind of filtering may be problematic if the remote end is multihomed, but for typical broker users, this would be the exception. There should also be filters for packets with IPv4-mapped addresses and some consideration for monitoring of the tunnels usage.

At least some statistics on bandwidth usage per tunnel might be useful. If somebody offers a broker to random Internet users, people should not be very surprised if they encounter DOS attacks. It is good if similar attacks and other problems can be detected, and such traffic blocked if necessary.

4 Dual Stack

A popular solution to allow an organisation to communicate over both IPv4 and IPv6 is to enable both protocols simultaneously, this is known as "dual-stack". New services can be easily installed as most modern equipment supports IPv6. The network administrators should remain cautious not to deploy IPv6 on an IPv4 equipment when the IPv6 filtering is not equivalent to the current IPv4 filtering.

It is possible for malicious users to access a private network by using IPv6 as an entry point, since basic filtering that should normally prevent such (as it should be for IPv4) is not often deployed. For an outsider to discover the IPv6 address of an important server may not be difficult, given that the system administrators tend to prefer allocating IPv6 address at the beginning of their prefix class (usually lower than ::ff), which makes the range of IPv6 addresses to scan for such a purpose a lot smaller. As a result, when connecting to a device running in dual-stack mode inside a network that was considered inaccessible by the outside world, it is likely without much effort to compromise further equipment inside this network (using either IPv4 or IPv6 from the entry machine).

In an incident, hackers, after they have compromised an IPv4 device, have enabled IPv6 in IPv4 tunnelling, with the aim to successfully bypass filtering and Intrusion Detection Systems (IDS) capabilities on the system for exchanging sensitive data.

5 6to4

6to4 [6to4] is an IPv6 over IPv4 tunnelling technique that has the capability of connecting isolated IPv6 networks to the IPv6 Internet. For the technique to work it requires each isolated network to deploy at least one 6to4 capable router that eventually has a connection to the IPv4 Internet. What makes 6to4 special is that the 6to4 router's IPv4 address used for the encapsulated packets is also used by definition to derive a unique /48 IPv6 prefix for addressing the hosts in its own network. Because of that one can use automatic tunnelling; when a 6to4 IPv6 address is learnt the IPv4 address of the tunnel end-point of the network it belongs to can also be derived.

In a 6to4 IPv6 network, the 6to4 router that encapsulates/decapsulates IPv6 packets is the only one required to support 6to4; the rest of the network uses pure IPv6. The associated IPv6 addresses begin with the standard 6to4 prefix 2002::/16 and are globally routable addresses in all respects. The 6to4 router will treat packets carrying these addresses in a special way though. When it forwards a packet with a 6to4 destination address, it encapsulates the packet into an IPv4 packet by deriving the IPv4 destination address from the 6to4 destination address. By this way IPv6 isolated IPv6 network islands can be interconnected via the existing IPv4 Internet infrastructure.

In order for an isolated island to deliver a packet to the main IPv6 Internet a 6to4 relay is required. The relay is a common 6to4 router that is willing to forward IPv6 packets to the rest of the IPv6 Internet, where the isolated islands have no IPv6 connectivity. The 6to4 router of an isolated island can then simply have an IPv6 default route to a 6to4 relay. The router will normally have a tunnel interface for 6to4 and a routing entry in the routing table for 2002::/16 pointing towards it. The relay will then be used when the destination address is not a 6to4 address. For IPv6 networks that are part of the IPv6 Internet to become able to reach the isolated IPv6 islands, they should internally announce a route towards the 6to4 relay; the relay can then forward the packets through the IPv4 infrastructure to the 6to4 router of the corresponding IPv6 island.

Tunnelling mechanisms, especially automatic ones, are always questionable from the security point of view (see chapter 2); if IPv4 addresses can be spoofed, anyone can inject any kind of traffic to the tunnel one wants -- and even if the source address spoofing is not possible, one may be able to launch several kinds of attacks.

Secure management of 6to4 is only an issue if using 6to4. As 6to4 is a transition mechanism, its use should be avoided if other alternatives, such as manually configured tunnelling or native IPv6 are available.

If 6to4 routers or relays are present, there are a lot of ways to attack or abuse them. However, it should be noted that in most cases, this is not any more attractive than abusing or attacking any other IPv4 (or IPv6) services.

The threats can be classified to several categories [Sec6to4]. These that can be easily protected against include:

1. attacks against the 6to4 pseudo-interface; these can be protected by adding an access-list to the pseudo-interface to filter out bad tunnelled packets:

```
a. deny from 2002::/16 to 2002::/16

b. allow from 2002::/16 to 2000::/3

c. deny everything else
```

In particular, c) denies link-local packets.

2. local directed broadcast attacks (on relays only); as above, an access list will handle this case.

For example assume the relay router has an interface with address 1.1.1.1/24.

Now, sending a packet to the relay which translates to sending a packet to

a broadcast address, here 1.1.1.255, would constitute a "local directed

broadcast attack". Similar is possible remotely if "no ip directed-broadcast" is configured.

These can be filtered e.g. by checking with an access list that when receiving packets, the 2002:0101:01ff::/48 address is not in the source or destination (depending on the broadcast addresses).

3. theft of service, if providing restricted 6to4 relay service; generally, this is not considered a big issue, but can be handled with careful routing policy and if necessary, access lists.

For instance, if the service is provided and advertised to only network operating under 1.1.0.0/16 and having 2001:db8::/32, then checks should be made that the encapsulated packets to 192.88.99.1 (if using the anycast relay service) would come from 1.1.0.0/16, and packets to 2002::/16 destinations would only come from 2001:db8::/32 or from 2002:0101::/32.

4. relay spoofing attack; 6to4 pseudo-interface is an interface like any other, and one must install usual access lists on it. For example, the site should configure the inbound access list to reject any source addresses that have been spoofed to belong to the site itself.

Consider a regular IPv6 site with 2001:db8:1::/48 prefix. The site's operators should install access list on its upstream interface similar to:

On input, to protect from someone sending packets from the Internet with this site's source address:

deny from 2001:db8:1::/48 to any

allow all

On output, to protect someone at the site sending packet out to the Internet with a wrong source address:

allow from 2001:db8:1::/48 to any

deny all

Similar checks must be installed on a 6to4 router for protection.

In addition, there are several methods how 6to4 can be used as to reflect a denial-of-service attack, to make it more difficult to trace. However, currently these problems are yet unresolved. Therefore, it is recommended to monitor the traffic levels of the 6to4 pseudo-interfaces regularly to see whether there are any anomalies and react if necessary.

6   ISATAP

This is an intra-site tunnelling technique that uses a site's IPv4 infrastructure as a virtual link layer. Normally when offering IPv6 connectivity within a site, one will need to have an IPv6 router on each link. With ISATAP [ISATAP] one only needs one IPv6 router with ISATAP support, and then all dual-stack hosts with ISATAP support can encapsulate IPv6 packets inside IPv4 and reach all other ISATAP hosts and routers. At each host one only needs to configure the address of an ISATAP router. There are also some ideas on how to autoconfigure this. Once this is done, normal IPv6 autoconfiguration can be done on the virtual link. The interface identifier will be based on the hosts IPv4 address. The IPv4 addresses are the link-layer addresses.

Since ISATAP uses tunneling, the general tunneling issues are valid. Another issue worth considering, is that all the site's ISATAP hosts will be on the same IPv6 link. It's very hard to protect someone on the link from someone else on the same link. Also there are several protocols that have some additional security by using link local addresses or hop limit (setting it to 255).

With no filtering in site border routers, the entire IPv4 Internet would be part of the link. It's important that the site does IPv4 and IPv6 ingress filtering. At the site border one should also drop IPv4 protocol 41 packets (IPv6 over IPv4 tunnels) unless they belong to known tunnels. There is still some spoofing possible inside the site, but that is comparable to the threat on a physical link. The problem might be somewhat bigger due to the potentially large number of hosts on the same link though. Except for the security aspects, there is no technical reason preventing the use of ISATAP between sites, but we highly recommend against it.

Monitoring traffic between the ISATAP hosts at a site is difficult. The hosts are on the same virtual link, so the packets do not pass through any routers (of course the packets might pass through IPv4 routers on the layer below). At the ISATAP router one can monitor traffic as usual though, but packets between hosts on the ISATAP link does not pass through the router.

7 DSTM

The DSTM [DSTM] mechanism is fully described in the 6NET cookbook D2.3.1.

7.1 Deployment procedures

?? DSTM server

The DSTM server is in charge of allocating a pool of IPv4 addresses. The system administrator has to decide this pool.

The IPv6 address of the Tunnel end-point (TEP) must be configured on the DSTM server.

?? Tunnel end-point

The tunnel end point must be connected to the IPv4 and the IPv6 Internet.

The IPv4 address pool that is managed by the DSTM server must be routed to the TEP.

?? Hosts

Every host in the network requires having an implementation of DSTM host. For the moment, there are implementations available for FreeBSD, Linux and Microsoft Windows 2000.

The hosts have IPv6 connectivity and therefore can have IPv6-only applications if there is no need to have IPv4 correspondents. For example, in the DSTM network, some internal applications can use only IPv6 and there is no need to support IPv4 anymore. Every application that will have IPv4 correspondents must support the dual IPv6-IPv4 stack.

The implementation provided by ENST for FreeBSD requires a "gif" (tunnel) interface to be available for an IPv4 over IPv6 tunnel.

?? The DSTM network

Once the DSTM mechanism is up and running, only IPv6 configuration of the network is necessary. The network monitoring is then much easier because the network administrator does not care about IPv4 anymore.

The DSTM service performance does not depend on the size of the DSTM network but on the number of people using DSTM at a time. This means that DSTM can be deployed on an important domain (for example an entire site or on an ISP network) at a stage when the amount of IPv4 traffic is low compared to IPv6. This means that DSTM mechanism will best fit to the needs there will be after an important IPv6 deployment.

7.2   Security recommendations

7.2.1   Control on who has access to the service

?? Filtering on the DSTM server

It is important for the network administrator to be sure that the hosts using the DSTM server are allowed to do so. The following filters should be deployed on the DSTM server:

```
allow ipv6 tunnel_request from allowed_hosts to DSTM_server

allow ipv6 tunnel_request from DSTM_server to TEP

allow ipv6 tunnel_reply from DSTM_server to allowed_hosts

allow ipv6 tunnel_delete from DSTM_server to allowed hosts

allow ipv6 tunnel_delete from DSTM_server to TEP

deny ipv6 from any to any
```

?? Authentication

For the time being, no authentication is performed, because the RCP based implementation does not support it. Future implementations that plan to use the Tunnel Setup Protocol (TSP) [TSP1][TSP2] and DHCPv6 [DCHCPv6] should support authentication.

7.2.2   DTI problems

The current implementation of DSTM provided by ENST requires that the DSTM server and TEP are on the same machine to have tunnels on the TEP coherent with the allocation made by the server. The implementation is mostly based on RPC [RPC]. At this time, there is no implementation of a communication protocol between the server and the TEP for tunnel setup, even if some ideas are foreseen (TSP and DHCPv6).

If the TEP is not physically located on the machine on which the DSTM server is installed, then it uses Dynamic Tunnel Interfaces (DTI) for tunnel setup. The mechanism is simple: when the TEP receives an IPv6 packet with protocol number 4 (meaning the data carried is an IPv4 packet), then it automatically creates an IPv4 in IPv6 tunnel using the information carried in the packet (IPv6 and IPv4 source and destination address). It is important to notice that there is no way in this case to check that the tunnels setup on the TEP corresponds to the allocations made by the DSTM server. The following case can happen: a host (A) can be allocated an IPv4 address and communicate to the IPv4 world using the IPv4 over IPv6 tunnel created using the DTI of the TEP. If another host (B) sends an IPv4 in IPv6 packet, using the same IPv4 source address as A, then the tunnel between the TEP and A will be destroyed and replaced for a tunnel between the TEP and B. This can be seen as a spoofing attack, and this is a reason why this approach is now obsolete.

7.2.3   Filtering policy in the network

All the equipment inside the IPv6 network where DSTM service is available must allow protocol 4 (IPv4) to allow IPv4 in IPv6 tunnelling.

7.3   Monitoring

For monitoring a DSTM enabled network, the system administrator should monitor the tunnels created on the TEP and the allocations made by the DSTM server. The system should be able to

check that the tunnels correspond to the allocations performed by the DSTM server and send alert messages in case an error occurs.

8 Tunnel Setup Protocol (TSP)

8.1 Introduction

There are some situations where it is not possible to use static tunnelling (when both end-points keep the same underlying addresses), for example:

?? Using a tunnel broker service, IPv4 hosts willing to receive an IPv6 address may connect every time with a different IPv4 address, because they are behind NAT or because they connect from different locations.

?? DSTM clients require dynamic IPv4 in IPv6 tunnels.

TSP is a protocol designed to exchange tunnel information, to make it possible to give an answer to the problems described above. TSP messages are exchanged on TCP port 4343.

8.2 Advantages of deploying TSP

The protocol makes it possible to authenticate the message sender (using MD5 as an example). This leads to many deployment possibilities:

8.2.1 DSTM

?? The DSTM server can check the tunnel requests come from allowed hosts.

?? The DSTM server can communicate the IPv4 in IPv6 tunnel characteristics to the DSTM client and the TEP. The TEP and the client can check the data comes from a well known server before configuring access.

The following scheme shows the different entities taking part in a TSP communication session. The TSP message flows are labelled as "tsp" arrows (tsp1 and tsp2) on the following scheme:
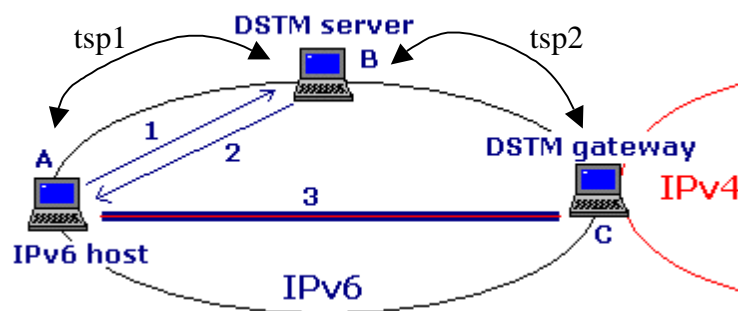

Figure 1: TSP and DSTM

8.2.2 Tunnel Broker

?? The broker can check whether the tunnel requests come from the correct host or router.

?? The broker TEP and the client can check whether the tunnel data come from a well-known broker.
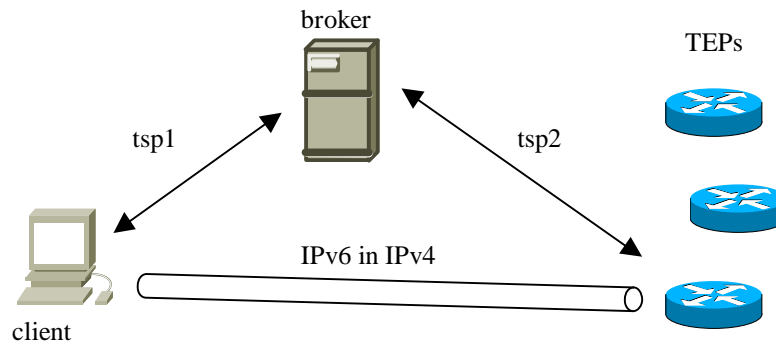
Figure 2: DSTM and tunnel broker

8.3 TSP implementations and deployment

Freenet6, one of the most important tunnel brokers in the Internet, has deployed TSP in their network; a lot of valuable information is provided via their website [FreeNet6]. They have decided to exploit the functionality provided by TSP in order to prevent abuse from users requesting many tunnels. Anonymous tunnels can be requested and are created with a lifetime of five days. By the time someone is registered with the broker they are given access to the service for three months.

Implementations of TSP clients are available on most operating systems (Microsoft Windows, Linux, BSD and others); no freely available implementation of a TSP server is known, but Freenet6 have already implemented theirs (though it is not publicly available).

ENST is to implement a DSTM client, server and TEP based on TSP. This makes it possible to have TEP configuration coherent with DSTM server allocations.

9 Detecting the presence of IPv6 in IPv4 tunnels

Networks containing IPv6-in-IPv4 tunnels are difficult to manage, because the tunnels are transparent to the rest of the network. In several scenarios, however, the ability to detect tunnels and determine their endpoints would be useful.

One example is troubleshooting: if a link in the tunnel's path fails, packets sent through the tunnel are lost, and an IPv6 traceroute will not reveal the source of the problem. In this case, the ability to determine that the failed link is in a tunnel, perhaps also performing an IPv4 traceroute between the tunnel endpoints, could help identify the faulty link and take appropriate action.

Troubleshooting is not the only scenario. For example, in some networks tunnels could be used as backup paths in case of problems with the main (native) links. In this case, it would be useful to know if a given destination is reached via a native link or via a tunnel, which may have lower performance. Or, a site or national network could use tunnels as an interim measure until native IPv6 infrastructure is in place; in such a network, a tool capable of detecting tunnels could determine how much still has to be done to complete the migration to native IPv6. Another example is managing dynamically generated tunnels, such as tunnels set up by a tunnel broker: it may be useful to determine the IPv4 address of a tunnel endpoint without having to query the tunnel broker application.

There are various methods for detecting the presence of a tunnel and/or determining the IPv4 addresses of the tunnel endpoints. Some of these suggest the presence of a tunnel, some suggest the possible IPv4 addresses of tunnel endpoints, and some do both. Examples are:

- IPv6 address inspection. For tunnels that embed the IPv4 address of the endpoint in the IPv6 address, such as 6to4 or ISATAP, inspection of the IPv6 addresses used may help deduce the presence of a tunnel and suggest the IPv4 address of one of the endpoints.

- Path MTU discovery. If path MTU decreases by 20 bytes between two consecutive nodes on a path, the link between the two nodes may be a tunnel. Path MTU discovery does not provide information on the endpoints, and in the case of multiple tunnels in a path, it detects only the first one.

- DNS lookups. If the DNS is appropriately configured, DNS mappings may provide information about tunnel endpoints. For example, if the DNS name of a tunnel interface maps to both an IPv6 address and an IPv4 address, the endpoint of the tunnel may be the IPv4 address.

- SNMP queries. The IP MIB and the tunnel MIB provide information on the presence of tunnels and their endpoints. The information gained through SNMP is usually accurate, but SNMP access is usually available only to network administrators.

Some of these methods yield accurate information, while others provide only hints. Also, some are always available, while others require administrative access to the network. Taking into account the fact that not all the methods provide the same information, a tool that could employ all of them, merging the results from each to form as accurate a hypothesis as possible, would be desirable. Such a tool would also be more efficient than manual intervention of an operator.

10 Conclusion

There are at the time many mechanisms to support the coexistence of IPv4 and IPv6. Most of them are already widely deployed (tunnelling, dual-stack) and the experience gained with them led to the compilation of the operational procedures described in this deliverable.

For every transition mechanism covered, network administrators can find recommendations regarding management and security issues raised by their deployment. The lack of security for some of the transition mechanisms has also been reported (e.g. DSTM dynamic tunnel interfaces) as the need of some supervision tools for the new features brought by the transition mechanisms. This document should be a basis for implementing and deploying such tools and add new management recommendations in the transition mechanisms' standardisation process.

11 References

[DSTM] "Dual Stack Transition Mechanism (DSTM)", J. Bound et al, IETF Internet Draft, December 2002, (draft-ietf-ngtrans-dstm-08.txt)

[RPC] "RPC: Remote Procedure Call Protocol Specification Version 2", R. Srinivasan, IETF RFC 1831, August 1995

[TSP1] "Tunnel Setup Protocol (TSP): A Control Protocol to Setup IPv6 or IPv4 Tunnels", M. Blanchet, IETF Internet Draft, July 2002, (draft-vg-ngtrans-tsp-01.txt)

[TSP2] "DSTM IPv4 over IPv6 tunnel profile for Tunnel Setup Protocol(TSP)", M. Blanchet, O. Medina, F. Parent, IETF Internet Draft, July 2002, (draft-blanchet-ngtrans-tsp-dstm-profile-01.txt)

[DHCPv6] „Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", R. Droms et al, IETF Internet Draft, November 2002, (draft-ietf-dhc-dhcpv6-28.txt)

[FreeNet6] "FreeNet6 web site", FreeNet6, http://www.freenet6.net, as accessed in January 2003

[Sec6to4] "Security Considerations for 6to4", P. Savola, IETF Internet Draft, January 2003, (draft-savola-v6ops-6to4-security-02.txt)

[KIN] "Requirements for Kerberized Internet Negotiation of Keys", M. Thomas, IETF RFC 3129, June 2001

[6to4] "Connection of IPv6 Domains via IPv4 Clouds", B. Carpenter, K. Moore, IETF RFC 3056, February 2001

[ISATAP] "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", F. Templin et al, IETF Internet Draft, January 2003, (draft-ietf-ngtrans-isatap-12.txt)

[NATPT] "Network Address Translation – Protocol Translation (NAT-PT)", G. Tsirtsis, P. Srisuresh, IETF RFC 2766, February 2000