Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 19, 2013

S. Steffann
S.J.M. Steffann Consultancy
I. van Beijnum
Institute IMDEA Networks
R. van Rein
OpenFortress

February 15, 2013

A comparison of IPv6 tunneling mechanisms
draft-steffann-tunnels-00

Abstract

This document provides an overview of various ways to to tunnel IPv6 packets over IPv4 networks. It covers mechanisms in contemporary use, touches on several mechanisms that are now only of historic interest, and discusses some newer tunneling mechanisms that are not (yet) widely used at the time of publication.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 19, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

During the transition from IPv4 to IPv6, IPv6 islands are separated by a sea of IPv4. Tunnels provide connectivity between these IPv6 islands. Tunnels work by encapsulating IPv6 packets inside IPv4 packets, as shown in the figure.

```
                                           +---------------+
                                           |     IPv4      |
                                           |    Header     |
                                           +---------------+
         +---------------+                 :   Optional    :
         |     IPv6      |                 : Encapsulation :
         |    Header     |                 :    Header     :
         +---------------+                 +---------------+
         |   Transport   |                 |     IPv6      |
         |    Layer      |      ===>       |    Header     |
         |    Header     |                 +---------------+
         +---------------+                 |   Transport   |
         |               |                 |    Layer      |
         ~     Data      ~                 |    Header     |
         |               |                 +---------------+
         +---------------+                 |               |
                                           ~     Data      ~
                                           |               |
                                           +---------------+
```

Encapsulating IPv6 in IPv4

Various tunnel mechanisms have been proposed over time. So many in fact, that it is difficult to get an overview.

Some tunnel mechanisms have been abandoned by the community, others have known problems and yet others have shown to be reliable. Some tunnel mechanisms were designed with a particular use-case in mind, others are generic. There may be documented limitations as well as limitations that have cropped up in deployment.

This document provides an overview of available and/or noteworthy tunnel mechanisms, with the intention to guide selection of the best mechanism for a particular purpose.

As such, the discussion of the different tunnel mechanisms is limited to the working principles of the different mechanisms and a few important details. Please use the references to learn the full details of each mechanism. The intended audience for this document is everyone who needs a connection to the IPv6 internet at large, but is not in the position to use native (untunneled) IPv6 connectivity,

and thus needs to select an appropriate tunneling mechanism. This document is also intended as a quick reference to tunnel mechanisms for the IETF community.

2. Terminology

Anycast: Mechanism to provide a service (in multiple locations) using multiple servers by configuring each server with the same IP address.

Dual stack: Also known as "dual IP layer". Nodes run IPv4 and IPv6 side by side, and can communicate with other dual stack nodes (over either IPv4 or IPv6), as well as IPv4-only nodes (over IPv4) and IPv6-only nodes (over IPv6). Most current operating systems are set up to use IPv4 when available as well as use IPv6 when available, allowing them to run in IPv4-only, IPv6-only or dual stack mode as circumstances permit. Except for a few things concerning the Domain Name System (DNS), there is no separate specification for dual stack beyond the specifications relevant to running IPv4 and IPv6. Dual stack is one of the three IPv4-to-IPv6 transition tools; the others are translation and tunnels.

Encapsulation: Transporting packets as data inside another packet. For instance, an IPv6 packet inside an IPv4 packet.

Host: A device that communicates using IP packets, but is not a router.

ISP: Internet Service Provider; the party connecting the outside of the local network's perimeter to the public Internet.

MTU: Maximum transmission Unit, the maximum size of a packet that can be transmitted over a link (or tunnel) without splitting it into multiple fragments.

NAT: Network Address Translation or Network Address Translator. NAT makes it possible for a number of hosts to share a single IP address. TCP and UDP port numbers are used to distinguish the traffic to/from different hosts served by the NAT; protocols other than TCP and UDP may be incompatible with NAT due to lack of port numbers. NAT also breaks protocols that depend on the IP addresses used in some way.

NBMA: Non-broadcast, multiple access. This is a network configuration in which nodes can exchange packets directly by addressing them at the desired destination. However, broadcasts or multicasts are not supported, so autodiscovery mechanisms such

as IPv6 Neighbour Discovery don't work.

Node: A device that implements IP, either a host or a router; also known as a system.

Path stretch: The difference between the shortest path through the network and the path (tunneled) packets actually take.

PMTUD: Path MTU Discovery, a method to determine the MTU between two systems where the traffic path may consist of multiple independent links. There are separate standards for PMTUD over IPv4 [RFC1191] and IPv6 [RFC1981].

Router: A device that forwards IP packets that it didn't generate itself.

System: A device that implements IP, either a host or a router; a node.

Translation: The IPv6 and IPv4 headers are similar enough that it is possible to translate between them. This allows IPv6-only hosts to communicate with IPv4-only hosts. The original specification for translating between IPv6 and IPv4, was heavily criticized by the Internet Architecture Board, but new specifications for translating between IPv6 and IPv4 were later published [RFC6145]. Translation is of the three IPv4-to-IPv6 transition tools; the others are dual stack and tunnels.

Tunnel: By encapsulating IPv6 packets inside IPv4 packets, IPv4-capable hosts and IPv6-capable networks isolated from other IPv6-capable systems or the IPv6 internet at large can exchange IPv6 packets over IPv4-only infrastructure. There are numerous ways to tunnel IPv6 over IPv4. This document compares these mechanisms. One of the three IPv4-to-IPv6 transition tools; the others are translation and dual stack.

Tunnel broker: A service that provides tunneled connectivity to the IPv6 internet, such as [SIXXS] and [TUNBROKER].

3. Tunnel Mechanisms

Automatic tunnels (Section 3.2) 6over4 (Section 3.3), 6to4 (Section 3.5), ISATAP (Section 3.7) and 6rd (Section 3.9) solve similar problems at different scales. They all encapsulate IPv6 packets immediately inside an IPv4 packet, without using additional headers. This is called "protocol 41 encapsulation" (see Section 5.1), as the Protocol field in the IPv4 header is set to 41

(decimal) to indicate that what follows is an IPv6 packet.

Each of these mechanisms also creates an IPv6 address for the host or router running the protocol based on the system's IPv4 address in one way or another (see Section 5.4). This lets 6to4, 6rd, ISATAP and automatic tunnels determine the IPv4 destination address in the outer IPv4 header from the IPv6 address of the destination, allowing for automatic operation without the need to administratively configure the remote tunnel endpoint.

6over4 and ISATAP provide IPv6 connectivity between IPv6-capable systems within a single organisation's network that is otherwise IPv4-only. 6rd allows ISPs to provide IPv6 connectivity to their customers over IPv4-only last mile infrastructures. 6to4 directly provides connectivity to the global IPv6 internet.

Configured tunnels (Section 3.1) also use protocol 41 encapsulation, but rely on manual configuration of the remote tunnel endpoint. Configured tunnels can be used within an organisation's network, but are typically used by tunnel broker services to provide connectivity to the IPv6 internet. GRE (Section 3.4) is similar to configured tunnels, but also supports tunneling protocols other than IPv6.

AYIYA (Section 3.6) is similar to configured tunnels and GRE, but typically uses a UDP header for better compatibility with NATs and is generally used with TIC (Section 4.1) to set up the tunnel rather than rely on manual configuration. Teredo (Section 3.8), 6a44 (Section 3.10) and 6bed4 (Section 3.11) are similar to 6to4, except that they are designed to work through NATs by running over UDP. Of these, Teredo assumes no ISP involvement and 6a44 does; and 6bed4 is designed to work over direct IPv4 paths between peers.

LISP (Section 3.12) is a system for abstracting the identifying function from the location function of IP addresses, which allows for the use of IPv6 for the former and IPv4 for the latter.

Please refer to Section 5 for more information about issues common to many tunnel mechanisms; those issues are not discussed separately for each mechanism. The mechanisms are discussed in chronological order of first publication below.

3.1. Configured Tunnels (Manual Tunnels / 6in4)

Configured and automatic tunnels are the two oldest tunnel mechanisms, originally published in "Transition Mechanisms for IPv6 Hosts and Routers" [RFC1933] in 1996. The latest specification of configured tunnels is "Basic Transition Mechanisms for IPv6 Hosts and Routers" [RFC4213], published in 2005. The mechanism is sometimes

called "manual tunnels" or "6in4".

Configured tunnels connect two systems in point-to-point fashion. The configuration that the name of the mechanism alludes to consists of a remote "tunnel endpoint". This is the IPv4 address of the system on the other side of the tunnel. When a system (potentially) has multiple IPv4 addresses, the local tunnel endpoint address may also need to be configured.

Due to their point-to-point nature, configured tunnels may carry multicast packets. As such, Neighbour Discovery can in principle operate over a configured tunnel. Configured tunnels use protocol 41 encapsulation.

The need to explicitly set up a configured tunnel makes them more difficult to deploy than automatic mechanisms. However, because there is a fixed, single remote tunnel endpoint, performance is predictable and easy to debug.

In the early days it was not unheard for a small network to get IPv6 connectivity from another continent. This excessive path stretch makes communication over short geographic distances much less efficient because the distance travelled by packets may be larger than the geographic distance by an order of magnitude or more.

Configured tunnels are widely implemented. Common operating systems can terminate configured tunnels, as well as IPv6-capable routers and home gateways. The mechanism is versatile, but is mostly used between isolated smaller IPv6-capable networks and the IPv6 internet, often through a "tunnel broker" such as tunnelbroker.net [TUNBROKER] or SixXS [SIXXS]. Before the existence of 6rd (Section 3.9), configured tunnels were also sometimes used by ISPs to connect their IPv6-capable customers across IPv4-only access infrastructure.

[RFC4891] discusses the use of IPsec to protect the confidentiality and integrity of IPv6 traffic exchanged over configured tunnels.

3.2. Automatic Tunneling

Automatic tunneling is described in [RFC2893], "Transition Mechanisms for IPv6 Hosts and Routers", but removed in [RFC4213], which is an update of RFC 2893. Configured tunnels (Section 3.1) are closely related to automatic tunnels and are specified in RFCs 2893 and 4213, too. Both use protocol 41 encapsulation.

Hosts that are capable of automatic tunneling use special IPv6 addresses: IPv4-compatible addresses. An IPv4-compatible IPv6 address consists of 96 zero bits followed by the system's IPv4

address. When sending packets to destinations within the IPv4-compatible ::/96 prefix, the IPv4 destination address in the outer IPv4 header is taken from the IPv4 address in the IPv4-compatible IPv6 destination address.

Automatic tunneling has a big limitation: it only allows for communication between IPv6-capable systems that both support automatic tunneling. There are no provisions for communicating with the native IPv6 internet. As such, the mechanism is of almost no practical use and is not implemented in current operating systems, as 6to4 (Section 3.5) does what automatic tunneling was supposed to do, but also provides connectivity to the rest of the IPv6 internet.

3.3. IPv6 over IPv4 without Explicit Tunnels (6over4)

[RFC2529], "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", was published in 1999. It's commonly known as "6over4".

6over4 is designed to work within a single organization's IPv4 network, where IPv6-capable hosts and routers are separated by IPv4-only routers. 6over4 treats the IPv4 network as a "virtual Ethernet" for the purpose of IPv6 communication. It uses IPv4 multicast to tunnel IPv6 multicast packets. A node's IPv4 address is included in the Interface Identifier used on the virtual 6over4 interface, allowing the exchange of protocol 41 encapsulated packets between 6over4 nodes without prior administrative configuration.

Because multicast is supported, standard IPv6 Neighbour Discovery and Stateless Address Autoconfiguration [RFC4862] can be used. Although like automatic tunnels (Section 3.2) and other mechanisms, 6over4 embeds the IPv4 address of the host is in the IPv6 address, the destination IPv4 address in the outer IPv4 header is *not* derived from the IPv6 address embedded in the inner IPv6 header, but learnt through Neighbour Discovery [RFC4861]. In effect, the IPv4 addresses of the hosts are used as link-layer addresses, in the same way that MAC addresses are used on Ethernet networks.

One or more routers with connectivity to the global IPv6 internet send out Router Advertisements to provide 6over4 hosts with connectivity to the rest of the IPv6 internet.

6over4 has the minimal protocol 41 encapsulation overhead and doesn't require manual configuration. 6over4 operation is stateless and peer-to-peer communication is supported within the IPv4 domain. Hosts can only take advantage of 6over4 if they run the mechanism themselves. 6over4 packets can't pass through a NAT successfully, as the IPv4 address exchanged through Neighbour Discovery will be different from the one needed to reach the host in question, and because without

port numbers, protocol 41 doesn't allow for multiplexing multiple hosts using this encapsulation behind a single IPv4 address. However, 6over4 works within IPv4 domains that use [RFC1918] addressing.

Because of its reliance on IPv4 multicast and because local IPv6 communication is relatively easy to facilitate using IPv6 routers, 6over4 is not supported in current operating systems, and should be considered obsolete. ISATAP (Section 3.7) provides very similar functionality without requiring IPv4 multicast capability, and is implemented in more operating systems.

3.4. Generic Routing Encapsulation (GRE)

Generic Routing Encapsulation (GRE) [RFC2784] is a generic point-to-point tunneling mechanism that allows many other protocols to be encapsulated in IP.

GRE is a simple protocol which is similar to 6in4 (Section 3.1) when used for IPv6-in-IPv4 tunneling. The main benefit of GRE is that is can not only encapsulate IPv6 packets but any protocol. The GRE header causes an extra overhead of 8 to 16 bytes depending on which options are used. GRE sets the Protocol field in the IP header to 47 (decimal).

The GRE header can optionally contain a checksum, a key to separate different traffic flows (for example different tunnels) between the same end points and a sequence number that can be used to prevent out of order packets to arrive.

GRE is implemented in many routers, but not in most consumer-level home gateways or desktop operating systems.

3.5. Connection of IPv6 Domains via IPv4 Clouds (6to4)

6to4 is specified in "Connection of IPv6 Domains via IPv4 Clouds" [RFC3056]. It creates a block of IPv6 addresses from a locally configured IPv4 address by concatenating that IPv4 address to the prefix 2002::/16, resulting in a /48 IPv6 prefix. Addresses in 2002::/16 are considered reachable through the tunnel interface, so the 6to4 network functions as a non-broadcast, multiple access (NBMA) network through which 6to4 users can communicate. IPv6 packets are encapsulated by adding an IPv4 header with the Protocol field set to 41 (decimal).

The /48 prefix allows a single system running 6to4 to act as a gateway or router for a large number of IPv6 hosts. Alternatively, an individual host may run 6to4 and not act as a gateway or router.

The system running 6to4 must have a globally reachable IPv4 address. Using a private IPv4 address [RFC1918] for 6to4 is not possile.

"An Anycast Prefix for 6to4 Relay Routers" [RFC3068] specifies an anycast mechanism for 6to4 relays that provide connectivity between the 6to4 network and the regular IPv6 internet. All public relays share the IPv4 address 192.88.99.1, which corresponds to 2002:c058:6301::. Relays advertise reachability towards 2002::/16 towards the native IPv6 internet, so packets addressed to systems using 6to4 addresses are routed to the closest gateway. The gateway encapsulates these packets and forwards them to the IPv4 address included in the IPv6 address. Systems running 6to4 have a default route pointing to 2002:c058:6301::, so they tunnel packets addressed to non-6to4 IPv6 destinations to the closest relay, which decapsulates the packet and forwards them as IPv6 packets

The 6to4 protocol adds minimal tunneling overhead (just the IPv4 header) and requires no manual configuration from the users. The biggest problem specific to 6to4 is that it is unpredictable which 6to4 anycast relay is used. These relays are often provided by third parties on a best-effort basis and do not always have enough bandwidth available. Traffic from the 6to4 network to the regular IPv6 internet will likely use a different 6to4 relay than the traffic in the opposite direction. If either of those relays is not reliable then the communication between those networks becomes unreliable. Especially the lack of control over the relay used for return traffic is considered to be a problem with 6to4.

For more information about 6to4, see the "Advisory Guidelines for 6to4 Deployment" [RFC6343].

Warning:

Although many, if not all, 6to4 implementations disable the mechanism when the system only has an RFC 1918 address, recently a block of IPv4 address has been set aside for use in service provider operated Network Address Translators, also known as Carrier Grade NAT (CNG). [RFC6598] sets aside the block 100.64.0.0/10 for the use between CGNs and subscriber devices. As 100.64.0.0/10 is not an RFC 1918 address block, systems implementing 6to4 may fail to disable the mechanism, but due to the shared nature of the 100.64.0.0/10 prefix, 6to4 cannot work using these addresses.

3.6. Anything In Anything (AYIYA)

[AYIYA] is designed for use by the [SIXXS] tunnel broker service. An Internet Draft was submitted [I-D.massar-v6ops-ayiya] but the process to make it an RFC was never completed.

The AYIYA protocol defines a method for encapsulating any protocol in any other protocol. The most common way of deploying AYIYA is to use the following sequence of headers: IPv4-UDP-AYIYA-IPv6, although other combinations like IPv4-AYIYA-IPv6 or IPv6-SCTP-AYIYA-IPv4 are also possible. The draft does not limit the contents nor the protocol that carries the AYIYA packets. In this document we only look at the most common usage (IPv4-UDP-AYIYA-IPv6) which is deployed on the SixXS tunnel brokers to provide IPv6 access to clients behind NAT devices.

AYIYA specifies the encapsulation, identification, checksum, security and certain management operations that can be used once the tunnel is established. It does not specify how the tunnel configuration parameters can be negotiated. Typically, the TIC protocol described in Section 4.1 protocol is used for that part of the tunnel setup, although the TSP protocol described in [RFC5572] could be used as well.

AYIYA provides a point-to-point tunnel, over which the endpoints can route traffic for any source and destination. When using SHA-1 hashing for authentication, as is common when using the AICCU client with a SixXS tunnel server, the total packet overhead is 72 bytes (20 for the IPv4 header, 8 for UDP and 44 for AYIYA).

AYIYA provides operational commands for querying the hostname, address, contact information, software version and last error message. An operational command to ask the other side of the tunnel to shut down is also available. These commands in the protocol can make debugging of AYIYA tunnels easier if the tools support them.

The main advantage of AYIYA is that it can provide a stable tunnel through an IPv4 NAT, and possibly multiple layers of NAT. The UDP port numbers allow multiple AYIYA users to reside behind a NAT. The client will contact the tunnel server at regular intervals and the tunnel server will automatically adapt to changing IPv4 addresses and/or UDP port numbers. The clients can be tracked through an (optional) identity field and (also optional) signature field. A timestamp is included in the AYIYA header to guard against replay attacks.

The main downside is that this protocol only seems to be in use by the [SIXXS] tunnel broker service and the [AICCU] client software.

3.7. Intra-site Automatic Tunnel Addressing (ISATAP)

ISATAP [RFC5214] uses protocol 41 encapsulation, to provide connectivity between isolated IPv6-capable nodes within an organisation's internal network. It is similar to 6over4

(Section 3.3), but without the requirement that the IPv4 network supports multicast. Unlike 6over4, ISATAP uses a Non-Broadcast Multiple Access (NBMA) communication model and thus doesn't support multicasts. The mechanism assigns IPv6 addresses whose interface identifier is solely defined by a node's IPv4 address, which is assumed to be unique.

In order to obtain a /64 prefix, an ISATAP tunnel endpoint needs to send a Router Solicitation. Without the ability to send and receive IPv6 multicasts, an ISATAP host must be configured with a Potential Router List through an all-IPv4 mechanism, such as manual setup, DHCP or the DNS. Site administrators are encouraged to use a DNS Fully Qualified Domain Name using the convention "isatap.domainname" (e.g., isatap.example.com). Hosts will accept packets with IPv4 sender addresses that are either on the Potential Router List, or that are embedded in the IPv6 sender address.

The router's prefix and the IPv4 address together define the IPv6 address for the ISATAP interface. This means that precisely one ISATAP address is available for each IPv4 address. As such, each host needs to run ISATAP itself in order to enjoy ISATAP IPv6 connectivity. The IPv4 address in the destination IPv6 address is used to bootstrap Neighbour Discovery.

[RFC5214] doesn't explicitly address the use of ISATAP using private [RFC1918] addresses. Despite that, the mechanism seems compatible with private addresses. NAT, however, breaks the relationship between the IPv4 address embedded in the IPv6 address and would therefore make communication between ISATAP hosts impossible. Any device that can communicate with the ISATAP hosts over IPv4 using protocol 41 can participate in the IPv6 subnet. It is therefore important to filter protocol 41 traffic at the network edge when NAT is not in use.

ISATAP is available in Windows as well as Linux. It is not recommended [ISATAP-WIN] for production networks running Windows if native IPv6 is available.

3.8. Tunneling IPv6 over UDP through NATs (Teredo)

Teredo [RFC4380] [RFC5991] [RFC6081] is designed as an automatic tunnel mechanism of last resort. It can configure an IPv6 address behind most NAT routers, but not all. Because Teredo uses encapsulation in UDP, multiple Teredo clients can be simultaneously active behind the same NAT router. For each Teredo client, a single IPv6 address is then created at the expense of a single external UDP port.

The operation of Teredo is based on a classification of NAT [RFC3489] as established during an interaction with a Teredo server. This classification has since been obsoleted [RFC5389] because it suggests more certainties about NAT than achieved in reality. Teredo however, relies on facilities induced from this classification, specifically the assumption that any NAT which is not classified as Symmetric NAT can receive a Teredo address because an external Teredo relay would be able to reach the Teredo client on the same external UDP port. This relay is selected near a native IPv6 destination address, so it must be dynamically switched during operation.

Teredo is present in Windows XP and later, and is enabled by default in Windows Vista and later. However, Windows will only use Teredo connectivity as a way to connect to IPv6 destinations of last resort, if no other IPv6 connectivity is present, Windows will not even look up AAAA records when resolving domain names. An open source implementation named Miredo exists for other platforms. This means that Teredo is only used to connect to explicit IPv6 addresses obtained through another mechanism than DNS.

The performance of Teredo falls noticeably short of that of IPv4. The setup time of a connection involves finding a Teredo relay nearby the native address to wrap and unwrap the traffic, and finding this relay can take in the order of seconds. This process is not sufficiently reliable; Teredo fails in about 37% [TERTST] of its attempts to connect to such native IPv6 peers. The roundtrip time of traffic can add tenths of a second, and jitter generally worsens if it is dependent on a public relay.

Teredo clients need to be configured with a Teredo server when setting up their local IPv6 address and when initiating a connection to a native IPv6 destination. The hostnames of the Teredo servers are usually pre-configured by the vendor of the Teredo implementation. All Microsoft Windows implementation use Teredo servers provided by Microsoft by default.

3.9. IPv6 Rapid Deployment (6rd)

6rd is specified in [RFC5969]. The original idea and the name come from [RFC5569] which described a successful "rapid deployment" of IPv6 by a commercial service provider. 6rd is used by service providers to connect customer networks behind a CPE to the IPv6 internet.

The structure of the 6rd protocol is based on 6to4 and it has the same minimal overhead as all protocols that use protocol 41 encapsulation. The main differences between 6rd and 6to4 are that 6rd is meant to be used inside a service provider's network and does

not use a special IPv6 prefix but one or more prefixes routed to the service provider. As such, 6rd users aren't recognisable by their IPv6 address like 6to4 users are. Where 6to4 uses (often public) relays based on global anycast routing 6rd uses relays provided and maintained by the service provider. Because of this architecture the tunnel does not traverse unknown networks which makes any debugging much easier.

6rd is completely stateless once it is configured. The tunnel endpoints can therefore be deployed using anycast. This is commonly done for the 6rd border relays deployed by the service provider to provide redundancy.

Because of the different prefix the device used as the 6rd client cannot use the hard-coded IPv6 prefix calculation and relay addresses of 6to4. Instead, the 6rd client needs to receive configuration information to work. In principle 6rd nodes may be configured in a variety of ways, but the most common one being through DHCP. If the client receives its IPv4 address from a DHCPv4 server then the 6rd configuration can be included in the DHCP message exchange using the 6rd DHCPv4 Option defined in [RFC5969]. Manual configuration of 6rd options and configuration using [TR-069] is also possible.

The main advantage of using 6rd is that it allows service providers to deploy IPv6 on core networks that for some reason cannot provide native IPv6 connectivity. It does not share the lack of predictable routing that 6to4 suffers from, because all routing, encapsulation and de-encapsulation is done by the service provider.

A disadvantage of 6rd for clients is that 6rd is only available when a service provider provides the relays and address space.

3.10. Native IPv6 behind NAT44 CPEs (6a44)

Inspired on Teredo, the 6a44 tunnel is described in "Native IPv6 behind IPv4-to-IPv4 NAT Customer Premise Equipment (6a44)" [RFC6751]. Its purpose is to enable Internet Service Providers to establish IPv6 connectivity for their customers, in spite of the use of a CPE or home gateway that is not prepared for IPv6. The infrastructure required for this is a 6a44 relay in the ISP's network and a 6a44 client in the customer's internal network.

6a44 was explicitly designed to overcome the noted problems with Teredo. Where Teredo was designed as a global solution without dependency on ISP co-operation, the 6a44 tunnel explicitly assumes ISP co-operation. Instead of using Teredo's well-known prefix, a /48 prefix out of the ISP's address space is used. A well-known (anycast) IPv4 address has been assigned for the 6a44 relay to be run

inside the ISP network without client configuration. This well-known address is allocated from the same IPv4 /24 as 6to4.

As part of its bootstrapping, a 6a44 client requests an address from the 6a44 relay, and a regular keepalive sent by the 6a44 client to the 6a44 relay keeps mapping state in NATs and firewalls on the path alive. Traffic passed from the native IPv6 internet to 6a44 is encapsulated in UDP and IPv4 by the relay and decapsulated by the 6a44 client; the opposite is done in the other direction.

The 6a44 protocol is very new, so it is not possible yet to give an overview of its operational impact. One detail that could be a cause for some concern is that the IPv6 addresses do not use the customary EUI-64 flags that normally signal a local address assignment strategy.

3.11. Peer-to-Peer IPv6 on Any Internetwork (6bed4)

The 6bed4 tunnel is specified in "6bed4: Peer-to-Peer IPv6 on Any Internetwork" [6BED4]. Unlike point-to-point tunneling mechanisms such as configured tunnels and AYIYA, 6bed4 also allows for direct communication between peers, similar to 6to4 and Teredo. The intent is to equal performance level of IPv4. It is currently an NBMA protocol; multicast may be supported in the future.

The setup of 6bed4 is reminiscent of 6to4, except that it employs UDP so it can be used behind NAT. It also has elements found in Teredo, but without a need to classify NAT and induce behaviour from that. The 6bed4 assumptions of NAT routers come down to plain vanilla UDP support. Given this, 6bed4 can create reliable IPv6 transports.

In environments where direct connections between 6bed4 peers is possible, additional path stretch compared to IPv4 communication is avoided, so 6bed4 performance comes close to IPv4 performance. In situations where this is not possible run over a the direct path between two peers because a NAT that does not conform to [RFC4787] is on the path, a fallback to a relay server is used. This increases path stretch and affects scalability through its impact on roundtrip times and jitter.

Another area where the relay is needed, is for connectivity between 6bed4 peers and native IPv6 hosts. For reasons of performance and scalability, connections between 6bed4 peers are preferred over connections between a 6bed4 peer and a native IPv6 host. A default address exists to support zero-config operation, but it is possible to send traffic out through a locally configured relay, which then also defines the relay for the return path.

6bed4 is the only tunnel today that is suitable for interactive media streams, provided that all media endpoints implement 6bed4, and prefer 6bed4-to-6bed4 traffic over 6bed4-to-native traffic. Under that premisse, the only hosts that need to go through a relay server are those that are behind a NAT with Address-Dependent Mapping or Address and Port-Dependent Mapping.

3.12. The Locator/ID Separation Protocol (LISP)

The Locator/ID Separation Protocol (LISP) [RFC6830] is a protocol to separate the identity of systems from their location on the internet and/or internal network. The addresses of the systems are called Endpoint Identifiers (EIDs) and the addresses of the gateways are called Routing Locators (RLOCs). It is possible to use IPv6 EIDs with IPv4 RLOCs and thereby use LISP for tunneling IPv6 over IPv4.

LISP defines its own packet formats for encapsulation of data packets and for control messages. All such packets are then encapsulated in UDP. Data packets use port 4341 and control packets use port 4342.

The LISP standard consists of several RFC documents. The relevant ones for this document are the basic standard [RFC6830], Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites [RFC6832] and the Locator/ID Separation Protocol (LISP) Map-Server Interface [RFC6833].

```
                    +----+    +----+
                    | MS |    | MR |
                    +----+    +----+       +-----+   /-----------\
                       |        |      /---| xTR |---| LISP site |
          +------+   /------------\---/    +-----+   \-----------/
          | PxTR |---| IP network |
          +------+   \------------/---\    +-----+   /-----------\
                            |          \---| xTR |---| LISP site |
                    /---------------\      +-----+   \-----------/
                    | Non-LISP site |
                    \---------------/
```

An example of a LISP deployment

LISP introduces new terminology and new concepts. The relevant ones for this document are:

ITR: Ingress Tunnel Router, a router encapsulating data packets at the border of a LISP site

ETR: Egress Tunnel Router, a router decapsulating data packets at the border of a LISP site

xTR: A router performing both the ITR and the ETR functions

PITR: Proxy ITR, a router accepting traffic from non-LISP sites, encapsulating it and tunneling it to the LISP sites

PETR: Proxy ETR, a router accepting traffic from LISP sites to send it to non-LISP sites

PxTR: A router performing both the PITR and the PETR functions

MS: Map Server, a server accepting RLOC registrations from ETRs

MR: Map Resolver, a server that can resolve queries for RLOCs from ITRs

LISP ETRs register the EID prefixes that they can handle traffic for in one or more Map Servers. ITRs and PITRs can then query Map Resolvers to determine which RLOCs to use when sending traffic to a LISP site. PITRs advertise aggregates of EID prefixes to the global routing table and provide tunneling services for them so that non-LISP sites can reach LISP sites. PETRs provide a way for LISP sites to send traffic to non-LISP sites.

LISP is a complex protocol if only used for tunneling. What it provides additionally is that ETRs can advertise their own RLOC addresses, that one site can have multiple xTRs with independent RLOCs and that the LISP site administrator can specify priorities and weights for those RLOCs. This provides redundancy and explicit load balancing between RLOCs. It also provides automatic tunneling between different sites without using a PxTR if both sites use Map Servers and Map Resolvers that are interconnected, for example by participating in the LISP Beta Network [LISPBETA].

4. Related Protocols

The following protocols are not tunneling mechanisms but they can be used in the configuration and/or setup phase of such protocols, or are otherwise relevant in the context of IPv6-in-IPv4 tunneling.

4.1. Tunnel Information and Control protocol (TIC)

The Tunnel Information and Control protocol (TIC) protocol [TIC] is a proprietary protocol for the [SIXXS] tunnel broker service.

With the TIC protocol a tunnel broker user can request a list of available tunnels and points-of-presence (POPs) from the tunnel broker service. When the user chooses one of the tunnels the configuration parameters for that tunnel can then be requested through TIC.

Authentication of users is done based on username and password. The only operational complexity is that a TIC node must have time synchronisation because TIC uses timestamps to avoid replay attacks.

4.2. Tunnel Setup Protocol (TSP)

The Tunnel Setup Protocol [RFC5572] is an experimental protocol for negotiating the setup of a variety of tunneling encapsulations. In this document we are only interested in the encapsulation of IPv6 in IPv4. The Tunnel Setup Protocol can negotiate these as a protocol 41 encapsulated tunnel or as a UDP encapsulated tunnel.

Tunnel negotiation is done with an XML exchange over UDP or TCP. The transport used for doing so may also be used as the IPv6 transport, but tunnel negotiation packets are marked to be distinguished.

When run over UDP, all general remarks for UDP-based tunnels apply. However, since a client exchanges all IPv6 traffic with the same tunnel server, there are no concerns related to the NAT implementation. The only concern is to send regular keepalives, for which ICMPv6 ping messages to the tunnel server are suggested.

When run directly over IPv4, all protocol 41 limitations apply. As such, the use of UDP is suggested unless there is a reason to prefer protocol 41 encapsulation.

However, the Tunnel Setup Protocol negotiates the IPv4 address of a client, but not its protocol and port. This is appropriate when protocol 41 is used, but for UDP it creates a situation where multiple users behind a NAT can claim the same tunnel access privileges. This is especially easy if v6anyv4 is negotiated over TCP. We therefore advise that clients should not use TCP for tunnel negotiation, and that servers should offer neither v6anyv4 nor v6udpv4 tunneling capabilities over TCP.

There are various security considerations related to TSP that are not mentioned in its RFC. A server supplies each client with an IPv6 address to use. The specification does not express concerns about tracking the relation between a client and their allocated IPv6 address; this is especially a concern when the IPv6 addresses are dynamically assigned.

Open source client software for the Tunnel Setup Protocol is available from the specification authors' freenet6 tunnel service. The same authors have not published a server in open source. A later, independent open source server implementation is incompatible with the clients because these clients do not adhere to the specification.

A public tunnel infrastructure is available by the name of gogo6, once again from the specification authors. As is common with centralised public tunnel infrastructure, this demonstrates the problem of scalability.

4.3. Dual-Stack Lite (Softwire)

Dual-Stack Lite [RFC6333], developed by the IETF Softwire working group, often comes up in discussions about IPv6 tunneling. However, Dual-Stack Lite (DS-Lite) is _not_ an IPv6-in-IPv4 tunneling mechanism; it is an IPv4-in-IPv6 tunneling mechanism.

DS-Lite allows ISPs to provide IPv4 connectivity over an IPv6-only access infrastructure. To this end, DS-Lite-capable home gateways encapsulate IPv4 packets inside IPv6 and forward them to a Carrier Grade NAT (CGN/CGNAT) device operated by the ISP. The CGN decapsulates the IPv4 packets, NATs them, and forwards them to the IPv4 internet.

IPv6 packets are handled through native IPv6 mechanisms and not tunneled.

5. General Issues

The following are aspects common to many or all tunneling mechanisms.

5.1. Protocol 41 Encapsulation

The most straightforward way to encapsulate an IPv6 packet inside an IPv4 packet is by simply adding an IPv4 header in front of the IPv6 header. In this case, the protocol field in the IPv4 header is set to the value 41 (decimal).

This simple protocol 41 encapsulation is used by a number of tunnel mechanisms:

configured tunnels (Section 3.1)

automatic tunneling (Section 3.2)

6over4 (Section 3.3)

6to4 (Section 3.5)

ISATAP (Section 3.7)

6rd (Section 3.9)

5.2. NAT and Firewalls

It is not uncommon for firewalls to block protocol 41 encapsulated packets, especially at the boundary between an organisation's internal network and the public internet. Non-proto-41 tunneling mechanisms typically employ a UDP header, and are somewhat less likely to be filtered.

Although protocol 41 can in principle work through NAT, there are two issues. First, when the IPv6 address is derived from the IPv4 address (see Section 5.4), NATting of the outer IPv4 header breaks the relationship between the IPv4 and IPv6 addresses. Second, because protocol 41 doesn't have any port numbers, only a single protocol 41 tunnel endpoint can be supported behind a NAT device with one IPv4 address (see Section 6.1). This limitation also applies to GRE.

Tunnels that pass through a NAT device or stateful firewall need to generate traffic at regular intervals to refresh the NAT or firewall mapping. If the mapping is lost, tunneled packets from the outside won't be able to pass through the NAT/firewall until a system behind the NAT or firewall sends a tunneled packet and the mapping is recreated. Alternatively, a static mapping (often in the form of a "default" or "DMZ" host) may be created.

The following tunneling mechanisms are incompatible with NAT:

automatic tunneling (Section 3.2)

6to4 (Section 3.5)

6rd (Section 3.9)

Note that it is common to run 6to4 or 6rd on a home gateway device that also performs IPv4 NAT. In this configuration, NAT is not applied to tunneled packets, so NAT and 6to4/6rd can coexist.

The following tunneling mechanisms cannot operate between nodes on opposing sides of a NAT, but they do work if _all_ nodes are behind a NAT and use RFC 1918 addresses:

6over4 (Section 3.3)

ISATAP (Section 3.7)

The following tunneling mechanisms may work through NAT in some circumstances, but are not designed for NAT compatibility:

configured tunnels (Section 3.1)

GRE (Section 3.4)

The following tunneling mechanisms are designed for NAT compatibility:

AYIYA (Section 3.6)

Teredo (Section 3.8)

6a44 (Section 3.10)

6bed4 (Section 3.11)

TODO:

LISP (Section 3.12)

A tunnel built over UDP makes a claim on a resource, namely an external UDP port. This may impact how well a tunnel will scale in an organisation; for instance, if every desktop runs its own tunnel client over UDP then the claim on this resource may have some impact.

Note that ISPs may have multiple subscribers share a public IPv4 address by performing NAT (Carrier Grade NAT, CGN or CGNAT in this context). In this case, the subscribers' home gateways may receive an address in the 100.64.0.0/10 block [RFC6598]. For the purposes of tunnling mechanisms, this address block is similar to the [RFC1918] address blocks. However, NAT/RFC1918 aware tunnel implementations may not recognise 100.64.0.0/10 as non-public addresses and fail to operate successfully.

5.3. MTU Considerations

Because of the the extra IPv4 header and possible additional headers between the IPv4 and IPv6 headers, tunnels experience a reduced

maximum packet size (Maximum Transfer Unit, MTU) compared to native IPv6 communication.

Path MTU discovery (PMTUD) should handle this in nearly all cases, but filtering of ICMPv6 "packet too big" messages may lead to an inability to communicate because senders of large packets fail to perform PMTUD successfully. However, when a tunnel terminates directly on the host using it, the TCP maximum segment size (MSS) option communicates the maximum packet size to the remote endpoint without relying on PMTUD.

With tunneling mechanisms where the MTU is left unspecified, it is not uncommon for the two endpoints to have different MTUs: typically, one uses the IPv6 minimum, 1280, while the other uses the physical MTU minus tunnel overhead, often 1480. In theory, this should lead to PMTUD failures because the "big" side unknowingly sends packets that the "small" side can't handle. However, in practice implementations handle incoming packets larger than their own MTU without issue.

Only when the IPv4 MTU is reduced below 1500 bytes, for instance when using PPP over Ethernet (PPPoE, [RFC2516]), issues are more likely to arise. With this in mind, it is prudent to set the MTU of a tunnel to no more than 1472 bytes, so tunneled packets can be transported over PPPoE links without fragmentation, or even 1280, to accommodate possible additional overhead.

5.4. IPv4 Addresses Embedded in IPv6 Addresses

Many tunneling mechanisms embed IPv4 addresses in the IPv6 addresses they use. There are two possible reasons for this. First, because the IPv4 address that needs to go in the outer IPv4 header can be derived from the destination IPv6 address, there is no need to explicitly configure tunnel endpoints. Automatic tunneling, 6to4, ISATAP and Teredo do this. 6over4 doesn't, but still embeds the IPv4 address in the interface identifier, and thus the IPv6 address, because that way, a (presumably) globally unique interface identifier can be generated.

Automatic tunneling uses IPv4-compatible addresses in the prefix ::/96 (i.e., the first 96 bits are all zero).

```
    |                     96 bits                    |        32       |
    +------------------------------------------------+-----------------+
    |                   0:0:0:0:0:0                  |  IPv4 address   |
    +------------------------------------------------+-----------------+
```

The IPv4-compatible addresses structure

Systems running 6to4 have addresses in the 6to4 prefix 2002::/16.

```
    |   16   |        32       |   16   |          64 bits             |
    +--------+-----------------+--------+------------------------------+
    |  2002  |  IPv4 address   | Subnet |        Interface ID          |
    +--------+-----------------+--------+------------------------------+
```

The 6to4 address structure

Because a 6rd domain might share a common IPv4 prefix it is not always necessary to encode all 32 bits of the IPv4 address in the 6rd delegated prefix. The bits that become available because of this optimisation can be used to provide more subnet IDs to the user and/or to use a smaller address block for the 6rd prefix.

```
    |     n bits    |    o bits    |   m bits  |    128-n-o-m bits     |
    +---------------+--------------+-----------+-----------------------+
    |  6rd prefix   | IPv4 address | subnet ID |     interface ID      |
    +---------------+--------------+-----------+-----------------------+
    |<--- 6rd delegated prefix --->|
```

The 6rd address structure

6over4 uses the IPv4 address to generate a 64-bit Interface Identifier, which can then be used to create a 128-bit IPv6 address through Stateless Autoconfiguration.

```
    |       48 bits       |   16   |        32       |        32       |
    +---------------------+--------+-----------------+-----------------+
    | Organisation prefix | Subnet |       0:0       |  IPv4 address   |
    +---------------------+--------+-----------------+-----------------+
```

The 6over4 address structure

The ISATAP address structure is similar to the 6over4 address structure, except that the unique/local (u) bit signifies whether the IPv4 address in the interface identifier is unique. Presumably, this is the case for any non-[RFC1918] IPv4 address. The group (g) bit is set to zero, and the remaining bits are set to to 0x00005EFE.

```
    |       48 bits       |   16   |        32       |        32       |
    +---------------------+--------+-----------------+-----------------+
    | Organisation prefix | Subnet |    ug00:5EFE    |  IPv4 address   |
    +---------------------+--------+-----------------+-----------------+
```

The ISATAP address structure

Teredo embeds the Teredo server's IPv4 address, a number of flags, a

UDP port number as well as the Teredo client's IPv4 address in the IPv6 addresses it creates. For good measure, the UDP port and client IPv4 address are "obfuscated" by flipping their bits.

```
    |     32 bits    |       32      |   16  |   16  |        32       |
    +----------------+---------------+-------+-------+-----------------+
    |     2001:0     |  Server IPv4  | Flags |  Port |   Client IPv4   |
    +----------------+---------------+-------+-------+-----------------+
```

The Teredo address structure

6a44 Can be seen as a combination of 6rd and Teredo. The 6a44 prefix is given out by an ISP. Both the customer site (home gateway) IPv4 address as well as the host's/client's RFC 1918 IPv4 address and also a port number are embedded in the IPv6 address.

```
    |        48 bits      |        32       |   16  |        32       |
    +----------------------+-----------------+-------+-----------------+
    |      6a44 prefix     | Cust. site IPv4 |  Port |   Client IPv4   |
    +----------------------+-----------------+-------+-----------------+
```

The 6a44 address structure

6bed4 embeds two combinations of an IPv4 address and UDP port (together acting as a "6bed4 address") in the IPv6 address; the first address is for a relay server that everyone is certain to reach, the other is for the direct address that most peers should be able to reach directly. The relay server however, is the only one with guaranteed access to the direct address.

```
    |  16    |        48 bits        |          50             |  14   |
    +--------+-----------------------+-------------------------+-------+
    | prefix | general 6bed4 address |  direct 6bed4 address   | lanIP |
    +--------+-----------------------+-------------------------+-------+
```

The 6bed4 address structure

The representation of the direct 6bed4 address is slightly modified to leave room in bits 70 and 71 for EUI-64 flags that signify that this local addressing scheme is used, and the unicast/multicast flag. The missing IPv4 address bits are moved to bits 112 and 113. The remaining 14 bits in the lanIP field can be used freely for local assignment.

6. Evaluation of Tunnel Mechanisms

The following subsections deal with the various aspects of tunnels

that guide their selection.

6.1. Efficiency of IPv4 Address Use

With the depletion of the IPv4 address space, the ability to deploy a tunnel mechanism behind NAT as well as the number of IPv6 subscribers, subnets and individual hosts that can be supported behind a single IPv4 address have become important considerations.

These issues are irrelevant to tunneling mechanisms that provide IPv6 connectivity between hosts within the same administrative domain, such as ISATAP or 6over4, as they can use private IPv4 addresses. This is also true for 6rd, which is used between an ISP and its customers' home gateways.

Although 6to4 can't work behind any kind of NAT and most other protocol 41 mechanisms can, at least in principle, in practice this difference is not as big, as the protocol 41 encapsulation doesn't provide any fields that allow a NAT to demultiplex tunneled packets. This means that only a single protocol 41 tunnel endpoint can be supported for each IPv4 address.

So a home or small office network can use 6to4 if the gateway has a public IPv4 address. A configured tunnel can also be terminated on a system that is behind a NAT, but only if no other systems attempt to use protocol 41 behind that same NAT (or rather, behind the same IPv4 address). This makes configured tunnels (as well as 6to4) incompatible with service provider operated NATs, where multiple subscribers share an IPv4 address. The same goes for GRE.

Teredo and 6bed4 are designed to work through NATs and use a UDP header, so multiple tunnel endpoints can be hosted behind a single IPv4 address. On the other hand, Teredo only provides IPv6 connectivity to a single host.

As such, we group IPv6-in-IPv4 tunneling mechanisms based on their IPv4 address use as follows, in order of declining IPv4 address use per IPv6 host:

One host: Automatic tunneling supports only a single IPv6 host per IPv4 address.

One SOHO: 6to4 can support a single home office or small office per IPv4 address.

One organisation: Configured tunnels and GRE can support one network, but of arbitrary size, behind an IPv4 address.

Many hosts: Teredo and 6bed4 support many individual hosts behind a single IPv4 address.

Many SOHOs: AYIYA can support many networks of arbitrary size behind a single IPv4 address. However, the need to maintain mapping state makes it less appropriate for networks larger than a home or small office network.

Not applicable: 6over4, ISATAP, 6rd and configured tunnels when used with RFC1918 addresses.

6.2. Supported Network Topologies

There are a few variations in the network topologies supported by IPv6 tunneling mechanisms. One aspect is whether it usually services a single host, a network or an ISP network. Another aspect is whether it supports multicast on the IPv6 level. Finally, a tunnel may be meant to connect to native addresses, or be suitable for direct traffic between peers on the same tunnel network.

Mechanism	Services	Multicast	Peering
Conf. tun.	Host/Network	Yes/No	N/A
Auto. tun.	Host	No	N/A
6over4	Network	Yes	N/A
GRE	Network	N/A	N/A
6to4	Network	No	Yes
AYIYA	Host	No	No
ISATAP	Host	No	Yes
Teredo	Host	No	No
6rd	ISP Network	N/A	N/A
6a44	Host	No	No
6bed4	Host	No	Yes
LISP			

Topologies Supported per Tunnel Mechanism

6.3. Parties Involved in Tunnel Realisation

Dependent on the mechanisms employed by a tunnel, more or less parties may have to be involved in setting up an IPv6 tunnel. This section details the parties that need to be willing to act before a tunnel can work.

Several tunnels require the presence of public gateways, usually at some well-known, anycasted address. Any particular instance of the gateway service may or may not provide a satisfactory service level, and the gateway used may be some distance away, adding path stretch. The gateway service must be available, and function properly if the tunnel is to work reliably and efficiently. Being dependent on a public gateway therefore incurs a risk of network delays. Public services are usually not under one's direct influence.

Other tunnels assume that an Internet Service Provider is involved in supplying the tunnel. This leads to more influence on the proper functioning of the tunnel, but it also makes the tunnel dependent on the selection of ISP.

The network perimeter filters between the ISP network and the local network usually contains firewalls and NAT. These components may be managed in part by the ISP. In general, the devices need to be co-operative for some tunnels to work reliably.

The local network may host relay servers for central tunnel management. In that case, a network administrator usually sets up such a node. Other tunnels are setup in local software, and could require an end user to have system administrative skills.

Mechanism	Management	Filtering Perimeter	ISP	Public Gateway
Conf. tun.	SysAdmin	Pass protocol 41	No	No
Auto. tun.	Automatic	Pass protocol 41	No	No
6over4	SysAdmin	Pass protocol 41	No	No
GRE	NetAdmin	Pass GRE	No	No
6to4	Automatic	No NAT	No	Yes (3)
AYIYA	NetAdmin	Pass plain UDP	No	Yes
ISATAP	SysAdmin	No NAT (1)	No	No
Teredo	Automatic	Problematic (2)	No	Yes
6rd	Automatic	Pass protocol 41	Yes	No
6a44	Automatic	Pass plain UDP	Yes	No
6bed4	Automatic	Pass plain UDP	No	Yes (4)
LISP				

Dependencies for Operating Tunnels

Notes:

(1): As an exception to the general rule that ISATAP is meant to run on public IPv4 addresses, ISATAP can be used to connect networks that are behind NAT if their address spaces may be united.

(2): Behind most NAT routers, Teredo should get an address allocated. It depends on the type of NAT if it will get through. This means that the protocol may be automatic, but the involvement of a NetAdmin may be required to make Teredo function.

(3): Normally, 6to4 is considered an automatic tunnel that sends to an anycast address in IPv4. For traffic returning from a native address to 6to4 space, another public service is required, and this one cannot be influenced by the sending party. Public service is not required for peer-to-peer traffic between 6to4 hosts.

(4): The public service is required to connect peers with an Endpoint-Dependent mapping in the direct path; furthermore, it is needed to connect 6bed4 to native addresses. When used to connect two 6bed4 peers, there is rarely a need for a public service.

6.4. Robustness

Tunnels may fail for three main reasons: when tunneled packets are filtered, typically by a firewall, when a tunnel endpoint IPv4 address changes, when tunneled packets are filtered or because of NAT issues.

If a tunnel endpoint gets a new address, the other side of the tunnel needs to know to send packets to the new address. With mechanisms that derive IPv6 addresses from the IPv4 address, the previous IPv6 addresses become unreachable and new IPv6 addresses must be configured.

Some tunneling mechanisms don't work through NAT, or are limited when working through NAT. NAT mappings can typically only be created by traffic from the "inside" to the "outside", not by traffic from outside the NAT to the network behind the NAT.

Point-to-point tunneling mechanisms either work consistently or they always fail. As such, a simple ping to the other side of the tunnel is sufficient to learn its state. Also, point-to-point tunnels may support routing protocols, which can automatically reroute traffic around a failed tunnel.

Some tunnel mechanisms use a public gateway to reach the native IPv6 internet. Public gateways may or may not be operational and/or reachable, and may have limited performance, depending on distance

and usage. Also, if multiple gateways are reachable at the same address (using an anycast setup), performance is hard to predict and debug. It is common for traffic in two directions to use different gateways, complicating debugging even further.

Tunnel mechanisms that use a broadcast or non-broadcast multiple access (NBMA) communication model may experience failures between some combinations of tunnel endpoints and not others.

Mechanism	Endpoint address change	Packet filtering	Public gateway	NAT mapping issues
Conf. tun.	failure	yes/no	no	yes (1)
Auto. tun.	interruption	per peer	no	N/A
6over4	interruption	per peer	no	N/A
GRE	failure	yes/no	no	N/A
6to4	interruption	gw, per peer	yes	N/A (2)
AYIYA	depends	yes/no	no	depends
ISATAP	interruption	per peer	no	N/A
Teredo	interruption	gw, per peer	yes	yes (3)
6rd	interruption	N/A	no	N/A
6a44	interruption	yes/no	no	no (4)
6bed4	interruption	yes/no	yes	no (4)
LISP				

Susceptibility of tunneling mechanisms to problems

Notes:

(1): only one protocol 41 tunnel endpoint can receive a NAT mapping behind a NAT using a single public IPv4 address. Additional endpoints will not receive incoming packets. When a tunnel endpoint changes its internal address, the old NAT mapping needs to time out before a new one can be created.

(2): 6to4 implementations automatically disable the mechanism when the system has an RFC 1918 address. However, 6to4 may remain enabled and be non-operational when ISPs apply NAT using non-RFC 1918 addresses [RFC6598].

(3): whether Teredo can obtain an address depends on the type of NAT it detects. Whether Teredo functions at such an address depends on the accuracy of that determination, which is founded in an incomplete model of NAT.

(4): 6a44 and 6bed4 make no assumptions about NAT, other than the standard ability to pass UDP out and then to be able for some time to receive return traffic from the same remote.

On some widely used implementations, 6to4 has been enabled by default without checking whether there was connectivity to the anycasted public gateway address. As a result, 6to4-derived connectivity to the IPv6 internet was often found to be broken because of protocol 41 filtering. Because of this, many operating systems now try to avoid using IPv6 over 6to4. See [RFC6343].

Also see [TERTST] for more information about the robustness of Teredo.

There is not a single tunneling mechanism that is more robust in all possible ways than every other tunneling mechanism. However, in general mechanisms that use public gateways and peer-to-peer tunneling tend to have the most issues. Configured tunnels on the other hand, often work very well, especially if there is no NAT on the path, but need administrative intervention when a tunnel endpoint address changes.

6.5. Performance

There are several reasons why tunneled connectivity may perform inferior to native, un-tunnteled connectivity. Inherently, tunnels add one or more extra headers, and therefore increase overhead. However, for a maximum size Ethernet packet the additional overhead of an IPv4 header is only 1.3%.

The process of encapsulation is not inherently slow, but in some implementations, it may be. Larger routers that normally forward packets using special purpose hardware, often don't have high performance CPUs. If then tunnel encapsulation must be done by that relatively slow CPU, performance will be worse than regular hardware-based packet forwarding.

The path that tunneled packets take can be longer than the path that untunneled packets would take. (Increased path stretch.) This may or may not lead to decreased performance.

Public gateways typically don't help performance. ISP-operated gateways are better.

Mechanism	Overhead (bytes)	Increased path stretch	Gateways	In practice	Variability
Conf. tun.	20	may be large	no	***	none
Auto. tun.	20	none	no	-	none
6over4	20	none	no	-	none
GRE	28 - 36	may be large	no	***	none
6to4	20	may be large	public	**	high
AYIYA	72	may be large	no	***	low
ISATAP	20	none	no	***	none
Teredo	28 ?	may be large	public	*	high
6rd	20	small	ISP	****	low
6a44	?	?		?	?
6bed4	?	?		?	?
LISP	?	small	ISP	?	high

Typical tunnel performance

7. IANA considerations

None.

8. Security considerations

There are many security considerations with tunneling. An important one is that through a tunnel, connectivity to the IPv6 internet may exist even though network administrators did not intend for it to be there. "Security Concerns with IP Tunneling" [RFC6169] discusses this issue in detail.

Another issue with tunneling is that it often makes implementation of ingress filtering (BCP 38, [RFC2827]) harder or even impossible. This is especially true for non-point-to-point tunnels, such as 6to4 and Teredo because legitimate packets can arrive from anywhere. So it is important to recognise that both IPv4 and IPv6 addresses in tunnel packets may be spoofed and cannot be relied upon for access controls.

Tunnels may also be used by third parties to obfuscate their activities or perform amplification attacks. As such, it is important to make sure only locally generated packets with legitimate addresses are sent out over tunnels.

9. Contributors

Job Snijders contributed text to the points of comparison.

10. Acknowledgements

We wish to thank SURFnet and Rogier Spoor for commissioning this work; both their initiative and funding has helped this document to be written.

11. References

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

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

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC2516] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D., and R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)", RFC 2516, February 1999.

[RFC2529] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529, March 1999.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[RFC2893] Gilligan, R. and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers", RFC 2893, August 2000.

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

[RFC3068] Huitema, C., "An Anycast Prefix for 6to4 Relay Routers", RFC 3068, June 2001.

[RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC4891] Graveman, R., Parthasarathy, M., Savola, P., and H. Tschofenig, "Using IPsec to Secure IPv6-in-IPv4 Tunnels", RFC 4891, May 2007.

[RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, March 2008.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5569] Despres, R., "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)", RFC 5569, January 2010.

[RFC5572] Blanchet, M. and F. Parent, "IPv6 Tunnel Broker with the Tunnel Setup Protocol (TSP)", RFC 5572, February 2010.

[RFC5969] Townsley, W. and O. Troan, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) -- Protocol Specification", RFC 5969, August 2010.

[RFC5991] Thaler, D., Krishnan, S., and J. Hoagland, "Teredo Security Updates", RFC 5991, September 2010.

[RFC6081] Thaler, D., "Teredo Extensions", RFC 6081, January 2011.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.

[RFC6169] Krishnan, S., Thaler, D., and J. Hoagland, "Security Concerns with IP Tunneling", RFC 6169, April 2011.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

[RFC6343] Carpenter, B., "Advisory Guidelines for 6to4 Deployment", RFC 6343, August 2011.

[RFC6598] Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space", BCP 153, RFC 6598, April 2012.

[RFC6751] Despres, R., Carpenter, B., Wing, D., and S. Jiang, "Native IPv6 behind IPv4-to-IPv4 NAT Customer Premises Equipment (6a44)", RFC 6751, October 2012.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

[RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, January 2013.

[RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, January 2013.

[I-D.massar-v6ops-ayiya]
Massar, J., "AYIYA: Anything In Anything", draft-massar-v6ops-ayiya-02 (work in progress), July 2004.

[TR-069] The Broadband Forum, "CPE WAN Management Protocol", July 2011, <http://www.broadband-forum.org/technical/download/TR-069_Amendment-4.pdf>.

[TUNBROKER]
Hurricane Electric, "Hurricane Electric Free IPv6 Tunnel

Broker", <http://www.tunnelbroker.net/>.

[SIXXS] Massar, J. and P. van Pelt, "IPv6 Deployment & Tunnel Broker", <http://www.sixxs.net/>.

[AYIYA] SixXS, "Anything In Anything (AYIYA)", <http://www.sixxs.net/tools/ayiya/>.

[AICCU] SixXS, "Automatic IPv6 Connectivity Client Utility (AICCU)", <http://www.sixxs.net/tools/aiccu/>.

[TIC] SixXS, "Tunnel Information and Control protocol (TIC)", <http://www.sixxs.net/tools/tic/>.

[TERTST] Huston, G., "Testing Teredo", April 2011, <http://www.potaroo.net/ispcol/2011-04/teredo.html>.

[ISATAP-WIN] Microsoft, "Intra-site Automatic Tunnel Addressing Protocol Deployment Guide", September 2010, <http://www.microsoft.com/en-us/download/details.aspx?id=18383>.

[6BED4] Van Rein, R., "6bed4: Peer-to-Peer IPv6 on Any Internetwork", <http://devel.0cpm.org/6bed4/>.

[LISPBETA] "LISP Beta Network", <http://www.lisp4.net/beta-network/>.

Appendix A. Evaluation Criteria

Each type of tunnel has specific advantages and disadvantages. We have considered the following points when evaluating the different protocols. Not every point is mentioned in each section where a protocol is described, only those that are specifically relevant to that protocol.

Protocol overhead: How much overhead does the tunneling protocol cause? There are two factors that play a role: number of interactions to set up the tunnel and packet header size causing a lower MTU and/or fragmentation.

Automatic configuration: Does this protocol require manual configuration at the endpoints?

Predictability: How predictable is the functioning of the protocol?

Single host or network: Is this protocol intended to be used by a single host or by a router that then provides IPv6 connectivity to multiple hosts?

Load balancing: Does the tunnel traffic have enough entropy and/or hashability to be able to be load-balanced over multiple links, or do all tunnel packets have the same outer 5-tuple?

Path stretch: Does the tunnel optimise the route, or is there a big potential for a much longer path when using the tunnel?

NAT traversal: Can the tunnel pass through a NAT gateway, and does it require configuration on that NAT gateway?

Tunnel endpoint mobility: Are the IPv4 addresses of the tunnel fixed or do they adjust automatically when an endpoint moves.

State: Are the endpoints required to keep state for the tunnel or is the tunnel stateless?

Network type: Is this network a point-to-point network, multipoint or NBMA type of network?

Purpose: What is the intended purpose of this tunnel protocol?

Related protocols: To which protocols is this tunnel protocol related? Are there alternatives?

Implementations: Is this protocol supported on the major operating systems, routers and firewalls?

Limitations: What are the known limitations of this protocol?

Authors' Addresses

Sander Steffann
S.J.M. Steffann Consultancy
Tienwoningenweg 46
Apeldoorn, Gelderland 7312 DN
The Netherlands

Email: sander@steffann.nl

Iljitsch van Beijnum
Institute IMDEA Networks
Avda. del Mar Mediterraneo, 22
Leganes, Madrid 28918
Spain

Email: iljitsch@muada.com

Rick van Rein
OpenFortress
Haarlebrink 5
Enschede, Overijssel 7544 WP
The Netherlands

Email: rick@openfortress.nl