Independent Submission   S. Steffann
Request for Comments: 7059   S.J.M. Steffann Consultancy
Category: Informational   I. van Beijnum
ISSN: 2070-1721   Institute IMDEA Networks
                                                            R. van Rein
                                                            OpenFortress
                                                           November 2013

            A Comparison of IPv6-over-IPv4 Tunnel Mechanisms

Abstract

   This document provides an overview of various ways to tunnel IPv6
   packets over IPv4 networks. It covers mechanisms in current use,
   touches on several mechanisms that are now only of historic interest,
   and discusses some newer tunnel mechanisms that are not widely used
   at the time of publication. The goal of the document is helping
   people with an IPv6-in-IPv4 tunneling need to select the mechanisms
   that may apply to them.

Status of This Memo

   This document is not an Internet Standards Track specification; it is
   published for informational purposes.

   This is a contribution to the RFC Series, independently of any other
   RFC stream. The RFC Editor has chosen to publish this document at
   its discretion and makes no statement about its value for
   implementation or deployment. Documents approved for publication by
   the RFC Editor are not a candidate for any level of Internet
   Standard; see Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7059.

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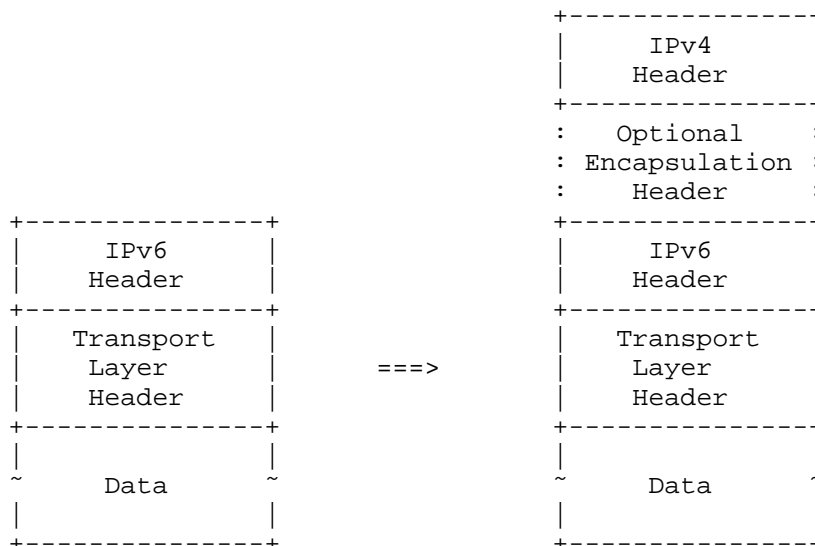
                      Encapsulating IPv6 in IPv4

   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. For brevity, only the most relevant documents are
   referenced. Refer to these for additional specifications, updates,
   and links to older versions of protocol specifications, as well as
   links to more general background information.

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 tunnel mechanism. However, when native IPv6
connectivity is available, this should be preferred over tunneled
connectivity as per rule 7 in Section 6 of [RFC6724]. This document
is also intended as a quick reference to tunnel mechanisms for the
IETF community.

The scope of this document is limited to tunnel mechanisms for
providing IPv6 connectivity over an IPv4 infrastructure. Mechanisms
for Virtual Private Networks (VPNs) and security architectures such
as IPsec [RFC4301], as well as IPv4-over-IPv6 tunneling, are out of
scope for this document as they serve a different purpose, even if
they could technically be used to provide IPv6 connectivity.

2. Terminology

   Anycast: Mechanism to provide a service, in multiple locations
      and/or using multiple servers, by configuring each server with the
      same IP address.

   Carrier-Grade NAT (CGN): A Network Address Translation (see NAT)
      device used by an ISP so multiple subscribers can be served using
      a single IPv4 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
      (using IPv4 or IPv6), as well as IPv4-only nodes (using IPv4) and
      IPv6-only nodes (using 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 a packet as data inside another packet.
      For instance, an IPv6 packet inside an IPv4 packet.

   Firewall: A device that selectively filters IP packets, allowing
      some protocols through but not others. A firewall may act as a
      switch, operating below the IP layer, or as a router.

   Host: A device that communicates using the Internet Protocol and
      only transmits packets from its own address.

   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. Typically, NAT devices behave as a
      host towards the public Internet, and as a router towards the
      internal network.

   NBMA: Non-Broadcast Multi-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 must be modified to use unicast to work.

   Node: A device that implements IP, either a host or a router; also
      known as a system. See note at "NAT".

   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 smallest MTU on
      the path between two nodes. There are separate specifications for
      PMTUD over IPv4 [RFC1191] and IPv6 [RFC1981].

   Router: A device that forwards IP packets that it didn't generate
      itself. See note at "NAT".

   System: A device that implements IP, either a host or a router; a
      network 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 criticised by
      the Internet Architecture Board, but new specifications for
      translating between IPv6 and IPv4 were later published [RFC6145].
      Translation is one of the three IPv4-to-IPv6 transition tools; the
      others are dual stack and tunnels.

   Tunnel: By encapsulating IPv6 packets inside IPv4 packets, IPv6-
      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.
      Tunneling is 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 [SIXXS], tunnelbroker.net
      [TUNBROKER], and gogo6 [GOGO6].

3. Tunnel Mechanisms

   Automatic tunnels (Section 3.2), configured tunnels (Section 3.1),
   6over4 (Section 3.3), 6to4 (Section 3.5), the Intra-Site Automatic
   Tunnel Addressing Protocol (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 to indicate that what follows is an IPv6 packet.

   6to4, 6rd, ISATAP, and automatic tunneling each generate an IPv6
   address or range of IPv6 addresses 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 without involving
   an ISP.

   Configured tunnels (Section 3.1) also use protocol 41 encapsulation
   but rely on manual configuration of the remote tunnel endpoint. (The
   Heartbeat Protocol (Section 4.2) solves this.) 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 encapsulating 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 the Tunnel Information and Control (TIC) protocol
   (Section 4.3) to set up the tunnel rather than rely on manual
   configuration. Teredo (Section 3.8), 6a44 (Section 3.10), and 6bed4
   (Section 3.13) are similar to 6to4, except that they are designed to
   work through NATs by running over UDP. Of these, Teredo and 6bed4
   assume no ISP involvement and 6a44 does; 6bed4 is designed to work
   over direct IPv4 paths between peers whenever possible.

   The Locator/ID Separation Protocol (LISP) (Section 3.11) is a system
   for abstracting the identifying function from the location function
   of IP addresses; this allows for the use of IPv6 for the former and
   IPv4 for the latter.

   The Subnetwork Encapsulation and Adaptation Layer (SEAL)
   (Section 3.12) and its companion technologies (Virtual Enterprise
   Traversal (VET), Asymmetric Extended Route Optimization (AERO),
   Internet Routing Overlay Network (IRON), and Routing and Addressing
   in Networks with Global Enterprise Recursion (RANGER)) provide a
   configured tunnel system for IPv6-in-IPv4 tunneling to default
   routers as well as automatic tunnel endpoint discovery for
   optimisation of more-specific routes.

   Dual-Stack Lite [RFC6333] and MAP [MAP], both developed by the IETF
   Softwire working group, often come up in discussions about IPv6
   tunneling. However, they are _not_ IPv6-in-IPv4 tunnel mechanisms.
   They are IPv4-in-IPv6 mechanisms for providing IPv4 connectivity over
   an IPv6 infrastructure.

   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 below in roughly
   chronological order.

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", "static tunnels", "protocol 41 tunnels", or
   "6in4".

   Configured tunnels connect two systems in point-to-point fashion
   using protocol 41 encapsulation. 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.

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

      In the early days, it was not unheard of 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], SixXS
      [SIXXS], or gogo6 [GOGO6].

      [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 a
      replacement 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 it also provides connectivity to the rest of the IPv6 Internet.

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

   "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels"
   [RFC2529] was published in 1999. The mechanism is commonly known as
   "6over4".

   6over4 is designed to work within a single organisation'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 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
   Media Access Control (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 overhead for protocol 41 encapsulation and
   doesn't require manual configuration. 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 reside behind a NAT in their entirety and
   use RFC 1918 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. ISATAP
   (Section 3.7) provides very similar functionality without requiring
   IPv4 multicast capability and is implemented more widely.

3.4. Generic Routing Encapsulation (GRE)

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

   GRE is a simple protocol that is similar to configured tunnels
   (Section 3.1) when used for IPv6-in-IPv4 tunneling. The main benefit
   of GRE is that it can encapsulate any protocol's packets, not only
   IPv6 packets. 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.

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

   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 Multi-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.

   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 6to4 with a private IPv4 address [RFC1918] is not possible.

   "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 to 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 overhead for protocol 41 encapsulation
and requires no manual configuration from 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. In practice, this has caused unpredictable
performance. 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 is affected.
Especially the lack of control over the relay used for return traffic
is considered to be a problem with 6to4.

To avoid problems with 6to4, the IPv6 Default Address Selection
algorithm [RFC6724] gives IPv4 addresses a higher preference than
6to4 addresses. When making a connection, a system will prefer
native IPv6 over IPv4, and IPv4 over 6to4 IPv6. This causes 6to4 to
be used only when a destination is not reachable over IPv4 and no
other IPv6 connectivity is available.

For more information about 6to4, see "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 addresses has been set aside for use in service-provider-
operated Network Address Translators, also known as Carrier-Grade
NATs (CGNs). [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. The same issue is
present if an ISP decides to use regular global unicast IPv4 address
space behind a CGN.

3.5.1. 6to4 Provider Managed Tunnels

[RFC6732] describes "6to4 provider managed tunnels", which are a way
to make 6to4 work behind a CGN. This is accomplished by running a
6to4 gateway at the 6to4 gateway anycast address, and then
translating the IPv6 addresses used by 6to4 users behind the CGN to
IPv6 addresses from the ISP's range. Unlike IPv4 NAT, where multiple

   internal hosts share a single public IPv4 address, prefix translation
   maps entire prefixes, so each host has its own public IPv6 address
   and can receive incoming packets as usual.

   However, if IPv6 applications are not aware that translation is
   happening (and they have no reason to expect that it is), they may
   not use their externally visible address in referrals, so
   applications that use referrals are likely to fail. Additionally,
   the translation is only specified for packets that flow through the
   6to4 gateway, not for packets sent directly to other 6to4 users. So,
   communication with other 6to4 users is not possible. As such, the
   use of 6to4 provider managed tunnels is discouraged except as a very
   last resort.

3.6. Anything In Anything (AYIYA)

   AYIYA [AYIYA] is designed for use by the SixXS [SIXXS] tunnel-broker
   service. For more information, see the specification [MASSAR].

   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. That document 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) that 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 4.3 is used for that part of the tunnel
   setup, although the Tunnel Setup Protocol (TSP) (Section 4.1) could
   just as well be used.

   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 Automatic
   IPv6 Connectivity Client Utility (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. The UDP port numbers allow multiple AYIYA users
   to share a single IPv4 address 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. To prevent a third party from injecting
   rogue packets into the tunnel, the client can optionally be
   authenticated by using the identity and signature fields. A
   timestamp is included in the AYIYA header to guard against replay
   attacks.

   There is currently a single implementation of this protocol: the
   AICCU [AICCU] client software used with the SixXS [SIXXS] tunnel-
   broker service.

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 Multi-
   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 host needs to send a
   unicast Router Solicitation to receive a unicast Router Advertisement
   from an ISATAP router. 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 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
RFC 1918 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.

ISATAP is available in Windows, Linux, and Cisco IOS.

3.8. Tunneling IPv6 over UDP through NATs (Teredo)

Teredo is specified in [RFC4380] and a few updates; it is designed as
an automatic tunnel mechanism of last resort. It can configure an
IPv6 address behind most NAT devices, but not all. Because Teredo
uses encapsulation in UDP, multiple Teredo clients can be
simultaneously active behind the same NAT. 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 (by [RFC5389]) because it
assigns more properties to NAT than achieved in reality.

Teredo is present in Windows XP and later and is enabled by default
in Windows Vista and later. However, if IPv6 connectivity is only
possible through Teredo, then Windows will not look up AAAA records
when resolving domain names. This means that Teredo is only used to
connect to explicit IPv6 addresses obtained through another mechanism
than DNS. An open-source implementation named Miredo exists for
other platforms.

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 encapsulate and decapsulate 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 native IPv6 destinations. 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 preconfigured by the vendor of the Teredo implementation.
All Microsoft Windows implementations use Teredo servers provided by
Microsoft by default.

3.9. IPv6 Rapid Deployment (6rd)

   6rd [RFC5969] is used by service providers to connect customer
   networks behind a CPE (Customer Premises Equipment) 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 immediately recognisable
   by their IPv6 address the way 6to4 users are. Where 6to4 uses 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; this 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, 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 last-mile access 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 decapsulation are done by the service provider.

6rd is intended to be a service managed by an ISP or enterprise IT
department that must explicitly make 6rd available for clients to be
able to use it.

3.10. Native IPv6 behind NAT44 CPEs (6a44)

Inspired by Teredo, the 6a44 tunnel is described in "Native IPv6
behind IPv4-to-IPv4 NAT Customer Premises 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 cooperation, the 6a44 tunnel explicitly assumes ISP
cooperation. 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.

3.11. 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 specification consists of several RFCs. The relevant ones
for IPv6-in-IPv4 tunneling are the base specification [RFC6830],
"Interworking between Locator/ID Separation Protocol (LISP) and Non-
LISP Sites" [RFC6832], and "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 for which they can handle traffic
   with 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. Features of
   LISP are 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 allows for 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]. To facilitate these
   interconnections, the LISP Delegated Database Tree (DDT) system is
   available.

   LISP is implemented on most Cisco devices. There are implementations
   available for FreeBSD and Linux, as well as a platform-independent
   implementation in the Python programming language. Note that for
   LISP to work, a mapping service not unlike the DNS must be in place.

3.12. Subnetwork Encapsulation and Adaptation Layer (SEAL)

   The Subnetwork Encapsulation and Adaptation Layer (SEAL) [SEAL]
   (along with its companion technologies cited therein) provides a
   hybrid configured/automatic tunneling system. SEAL itself provides a
   mid-layer of encapsulation between the inner IPv6 header and the
   outer IPv4 header, i.e., as IPv4-SEAL-IPv6. SEAL can also be used in
   conjunction with an outer UDP encapsulation header, e.g., if NAT
   traversal is necessary.

   The SEAL tunnel endpoint creates bidirectional configured tunnels to
   reach default IPv6 routers, and it discovers unidirectional automatic
   tunnels. SEAL tunnels can be configured over multiple underlying
   IPv4 links whether the addresses are provisioned from public or
   private IPv4 addressing domains. In that case, multihoming and
   traffic engineering are naturally supported.

   SEAL provides an optional 32-bit identifier and variable-length
   Integrity Check Vector that can be used for packet identification,
   message origin authentication, anti-replay, and a mid-layer
   segmentation and reassembly capability. SEAL also provides a SEAL
   Control Message Protocol (SCMP) used for neighbour coordinations
   between tunnel endpoints. These coordinations are used for functions
   such as tunnel MTU signalling, route optimisations, neighbour
   reachability testing, and so on.

   SEAL ensures that packets that are no larger than 1500 bytes can be
   transported through the tunnel by using a tunnel segmentation
   function. IPv6 packets that are too large to transport through the
   tunnel whole are split into segments. The segments are encapsulated
   in IPv4 and reassembled into the original IPv6 packets at the remote

tunnel endpoint. SEAL also admits packets larger than 1500 bytes
into the tunnel on a best-effort basis in case the path between the
tunnel endpoints can support the larger size.

When SEAL is used alone without its companion technologies, it can be
used in the same scenarios as for GRE. However, SEAL provides
advanced capabilities that make it better suited than GRE for many
use cases. There is currently an experimental open-source
implementation of SEAL.

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

The 6bed4 tunnel is specified in "6bed4: Peer-to-Peer IPv6 on Any
Internetwork" [6BED4]. A specific goal of 6bed4 is to achieve direct
communication between peers when the intermediate infrastructure does
not prohibit it. The advantage of direct communication is to get a
performance level similar to IPv4. The address of a 6bed4 peer is
formed from the publicly visible IPv4 address and UDP port. The
tunnel service used for fallback connectivity can run anywhere --
perhaps at the local ISP or with a third-party service provider for
6bed4, or even on a well-known address. It is currently an NBMA
protocol; there are openings for expansion with multicast.

The setup of 6bed4 is somewhat similar to 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 NATs and induce behaviour
from that. The 6bed4 tunnel makes no assumptions about the
capabilities of NAT devices beyond being able to do straightforward
NAT on UDP packets. Given this, 6bed4 can create reliable IPv6
tunnels.

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

Another area where the tunnel server 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 locally configure a tunnel server as a fallback
route, which then also defines the tunnel server for the return path.

6bed4 has been specifically designed to support real-time interactive
traffic streams, such as SIP calls, between 6bed4-supporting end
points, assuming that each prefers 6bed4-to-6bed4 traffic over 6bed4-
to-native traffic. Under that premise, the only hosts that need to
go through a tunnel server are those that are behind a NAT with
Address-Dependent Mapping or Address and Port-Dependent Mapping. A
number of different implementations of 6bed4 have been constructed
[6BED4] during the ongoing development of its specification.

4. Related Protocols

   The following protocols are not tunnel mechanisms, but they can be
   used in the configuration and/or setup phase of such protocols.

4.1. Tunnel Setup Protocol (TSP)

   The Tunnel Setup Protocol [RFC5572] specifies a protocol for
   negotiating the setup of a variety of tunnel 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. The tunnel
   negotiation is performed as an XML exchange over UDP or TCP.

   As a TSP client exchanges all IPv6 traffic with the same tunnel
   server, there are no concerns caused by NAT implementations. The
   only concern is to send regular keepalives, for which ICMPv6 ping
   messages to the tunnel server are suggested. When encapsulating IPv6
   packets directly in IPv4, all protocol 41 limitations apply. To
   avoid these, an additional UDP header may be used.

   The Tunnel Setup Protocol treats all protocols and ports for one IPv4
   client address as equivalent. This suffices when protocol 41 is
   used, but for UDP it creates a situation where one user can set up a
   tunnel behind NAT, and another user could hijack the tunnel
   privileges.

   Open-source clients for the Tunnel Setup Protocol and a matching
   tunnel infrastructure are provided from the Freenet6 tunnel service
   [GOGO6].

4.2. SixXS Heartbeat Protocol

   The SixXS Heartbeat Protocol [HEARTBEAT] allows nodes that have
   intermittent connectivity or a dynamic IPv4 address that changes from
   time to time to have continuing tunneled IPv6 connectivity. One of
   the goals of the protocol is to determine when a node is no longer
   present at its previous IPv4 address and then stop sending tunneled
   packets to prevent them from being delivered to the wrong node. The

Heartbeat Protocol then allows a tunnel broker to determine a
client's new IPv4 address and continue sending tunneled packets with
minimal interruption.

To accomplish this, a node sends periodic heartbeat packets to the
tunnel broker. If the tunnel broker fails to receive valid heartbeat
packets, it shuts down the tunnel in question. Heartbeat packets
contain an MD5 message authentication code and a timestamp to avoid
replay attacks. The Heartbeat Protocol can work with different
tunnel mechanisms, but it is often used with configured tunnels
(Section 3.1).

The protocol is implemented in the SixXS tunnel broker; client
implementations are available for common operating systems. AYIYA
can be considered the successor of the Heartbeat Protocol.

4.3. Tunnel Information and Control Protocol (TIC)

The Tunnel Information and Control (TIC) protocol [TIC] is the setup
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. TIC also provides commands to control the tunnel, for
example, to change the tunnel endpoints or to enable or disable the
tunnel.

Authentication of users is done based on username and password.
Certain tunnel mechanisms, such as AYIYA (Section 3.6) and configured
tunnels (Section 3.1) with heartbeat (Section 4.2), need a
synchronised clock between the tunnel server and the client. TIC
facilitates this by providing a server timestamp on request. The
client can use that to verify that its clock is synchronised with the
server and show an error message to the user if it is not.

The TIC protocol is implemented in the AICCU client software (see
[AICCU]) and in AVM Fritz!Box home routers.

5. Common Aspects

   The following are aspects common to many or all tunnel 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. This encapsulation is also known as "IPv6 in IPv4"
   and "IP6 Encapsulation".

   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 NATs and firewalls to block protocol 41
   encapsulated packets, especially at the boundary between an
   organisation's internal network and the public Internet. Tunnel
   mechanisms that don't use protocol 41 encapsulation typically employ
   a UDP header and are somewhat less likely to be filtered, assuming
   that tunneling is initiated on the LAN side. UDP is usually
   subjected to Network Address Port Translation (NAPT) [RFC2663], which
   additionally translates between internal and external port numbers.
   (Often, the term "NAT" is used where "NAPT" may be more appropriate.)

   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), NATing of the outer IPv4 header breaks the
   relationship between the IPv4 and IPv6 addresses. Second, because
   protocol 41 does not use port numbers, the number of protocol 41
   tunnel endpoints that can be supported behind a NAT device is equal
   to its number of external IPv4 addresses (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 or firewall until a system
behind the NAT or firewall sends a tunneled packet and the mapping is
re-created. Alternatively, a static mapping (often in the form of a
"default" or "DMZ" host) may be configured manually.

The following tunnel mechanisms are incompatible with NAT because
their addresses must be derived from a globally unique IPv4 address:

    automatic tunneling (Section 3.2)

    6to4 (Section 3.5)

    6rd (Section 3.9)

    LISP (Section 3.11)

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 tunnel mechanisms cannot operate between nodes on
opposing sides of a NAT, but they do work if _all_ nodes are behind a
NAT (where RFC 1918 addresses are often used):

    6over4 (Section 3.3)

    ISATAP (Section 3.7)

The following tunnel 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 tunnel mechanisms are designed for NAT compatibility:

    AYIYA (Section 3.6)

    Teredo (Section 3.8); but it is unreliable

    6a44 (Section 3.10)

      SEAL (Section 3.12)

      6bed4 (Section 3.13)

   The LISP specification requires that locator addresses (the addresses
   in the outer IPv4 header) are globally routable public addresses.

   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 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 tunnel
   mechanisms, this address block is similar to the RFC 1918 address
   blocks. However, tunnel implementations that are aware of NAT and
   RFC 1918 addresses may not recognise 100.64.0.0/10 as non-public
   addresses and fail to operate successfully. The same issue is
   present if an ISP decides to use regular global unicast IPv4 address
   space behind a CGN.

5.3. MTU Considerations

   Because of the extra IPv4 header and possible additional headers
   between the IPv4 and IPv6 headers, tunnels experience a reduced
   maximum packet size (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,
   so TCP-based communication may still succeed. If not, the initial
   TCP SYN/ACK exchange happens without issue, but then the session
   stalls as the larger packets containing data are lost.

   With tunnel mechanisms where the MTU is left unspecified, it is
   possible for the two endpoints to have different MTUs: typically, one
   uses the IPv6 minimum (1280 bytes), while the other uses the physical
   MTU minus tunnel overhead (often 1480 bytes). 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. So, when the possibility exists that tunneled
packets encounter a PPPoE link, it is prudent to set the MTU of a
tunnel to no more than 1472 bytes, so that tunneled packets don't
have to be fragmented. Additionally, Section 3.2.1 of [RFC4213]
recommends limiting the MTU of tunnels to a minimum of 1280 bytes.

SEAL was specifically designed to overcome these limitations by
adding the capability to fragment IPv6 packets prior to encapsulation
in IPv4 and then to reassemble the fragments at the remote tunnel
endpoint. This way, the SEAL tunnel ensures that packets that are no
larger than 1500 bytes will be transported to the tunnel far end even
if there are restricting links in the path. SEAL can also admit
larger packets into the tunnel on a best-effort basis in case the
path between the tunnel endpoints can support this larger size.

5.4. IPv4 Addresses Embedded in IPv6 Addresses

Many tunnel mechanisms embed IPv4 addresses or further information in
the IPv6 addresses they use. There are two possible reasons for
this. First, with an IPv4 address embedded in the IPv6 address, the
outer IPv4 header can be derived without a need to explicitly
configure tunnel endpoints. Automatic tunneling, 6to4, ISATAP,
6bed4, and Teredo do this. 6over4 embeds the IPv4 address for the
second reason; it is embedded 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 Address Structure

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

```
| 16 bits |       32       |   16   |              64            |
+---------+----------------+--------+----------------------------+
|  2002   |  IPv4 address  | Subnet |        Interface ID        |
+---------+----------------+--------+----------------------------+
```
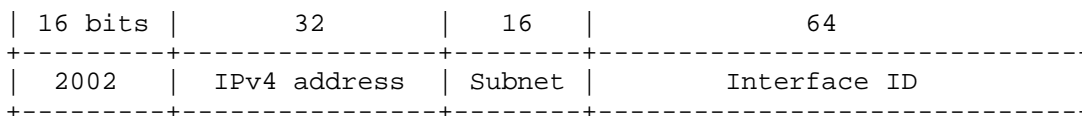
                 The 6to4 Address Structure

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 prefix  | Subnet ID|      Interface ID     |
+---------------+--------------+----------+-----------------------+
|<--- 6rd delegated prefix --->|
```
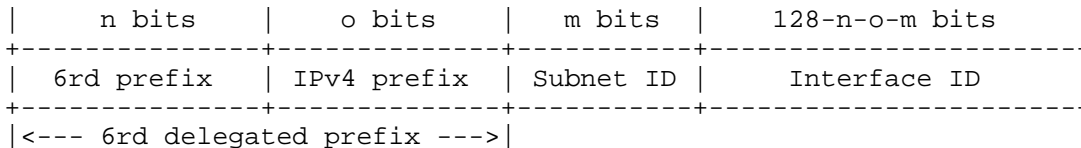
                     The 6rd Address Structure

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 Address Autoconfiguration.

```
|        48 bits        |   16   |       32       |       32       |
+----------------------+--------+----------------+----------------+
| Organisation prefix  | Subnet |      0:0        |  IPv4 address  |
+----------------------+--------+----------------+----------------+
```
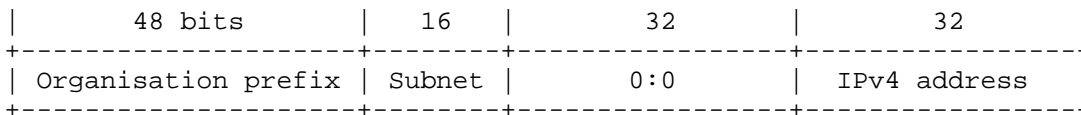
                    The 6over4 Address Structure

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 IPv4 address that is not as defined in RFC 1918.
The group (g) bit is set to zero, and the remaining bits are set to
0x00005EFE.

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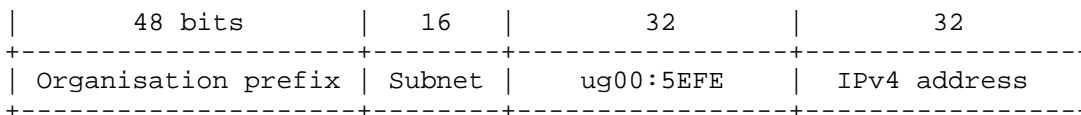
                    The ISATAP Address Structure

Teredo embeds the Teredo server's IPv4 address, a number of flags,
and 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 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 tunnel server that everyone is certain to
reach; the other is for the direct address that most peers should be
able to reach directly. The tunnel server, however, is the only one
with guaranteed access to the direct address.

```
| 32 bits |         32          |           50            |  14  |
+----------+--------------------+------------------------+-------+
|  prefix  |general 6bed4 address|  direct 6bed4 address   | lanIP |
+----------+--------------------+------------------------+-------+
```

                      The 6bed4 Address Structure

The general 6bed4 address field conceals the well-known UDP port for
the 6bed4 service. The direct 6bed4 address field includes two extra
bits to adhere to the EUI-64 address format. The lanIP bits are free
for local purposes, such as creating a DHCPv6 range.

6. Evaluation of Tunnel Mechanisms

   The following subsections deal with 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 tunnel 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; it is used between an ISP and its
   customers' home gateways when the ISP has implemented NAT.

   6to4 cannot work behind any kind of NAT. Most other mechanisms based
   on protocol 41 can work behind NAT, at least in principle. In
   practice, this difference is not as big because 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 public IPv4 address.

   This makes configured tunnels (as well as 6to4) incompatible with
   service-provider-operated NATs, where multiple subscribers share an
   IPv4 address.

   Teredo, 6a44, 6bed4, AYIYA, SEAL, and TSP 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.

   The following table shows how many IPv4 addresses each tunnel
   mechanism requires and how many IPv6 hosts it can connect. The
   mechanisms are listed in order of increasing numbers of supported
   IPv6 hosts per IPv4 address.

```
+------------+------------+------------+------------+------------+
| Mechanism  | Tunnels per| IPv6 hosts | Public IPv4| NAT        |
|            | IPv4 addr. | per tunnel | address    | compatible |
+------------+------------+------------+------------+------------+
| Auto. tun. | one        | one        | required   | no         |
| 6to4       | one        | multiple   | required   | no         |
| LISP       | one        | multiple   | required   | no         |
| 6rd        | one        | multiple   | not needed | no         |
| Conf. tun. | one        | multiple   | not needed | limited    |
| GRE        | one        | multiple   | not needed | limited    |
| Teredo     | multiple   | one        | not needed | yes (*)    |
| 6bed4      | multiple   | multiple   | not needed | yes        |
| 6a44       | multiple   | multiple   | not needed | yes        |
| AYIYA      | multiple   | multiple   | not needed | yes        |
| SEAL       | multiple   | multiple   | not needed | yes        |
+------------+------------+------------+------------+------------+
```

              Tunneled IPv6 Hosts per IPv4 Address

   (*) Although Teredo is designed for NAT compatibility, it doesn't
       work through all existing NATs.

6.2. Supported Network Topologies

   There are two ways to use an IPv6-in-IPv4 tunnel to connect to the
   IPv6 Internet: using a point-to-point tunnel to a tunnel broker or an
   ISP-operated gateway, or using a Non-Broadcast Multi-Access (NBMA)
   tunnel and anycasted public gateways or relays.

   The advantages of the point-to-point model are predictable
   performance and flexibility regarding the IPv6 addresses used. The
   advantage of the NBMA model is that traffic between two hosts or
   networks that both use the mechanism can flow directly without
   passing through a gateway (direct peer-to-peer communication). An
   extra advantage of the NBMA model with public gateways is automatic
   configuration and no involvement from an ISP.

   Unfortunately, the advantages of this NBMA public anycast model come
   at a price: both the peer-to-peer connectivity between tunnel users
   and the connectivity towards the native IPv6 Internet may suffer from
   reliability and performance issues.

   The anycast mechanism allows tunnel users to utilise the nearest
   gateway to connect to the IPv6 Internet by simply giving each gateway
   the same address. Routing protocols then select the lowest-cost (and
   presumably, shortest) path towards a gateway. However, this makes
   the path taken by tunneled packets hard to predict or influence. It
   is common for traffic in two directions to use different gateways,

complicating debugging even further. Because nobody is in charge or
gets paid for operating a gateway, the number of public gateways is
lower than would be ideal. This increases the distance to the
nearest gateway for some users. There is also the possibility that
gateways encounter more traffic than they can handle.

The advantage of a tunnel provided by an ISP or tunnel broker is that
there is a clear responsibility for providing a good service with
well-maintained gateways.

```
+------------+--------------+------------------------------+
| Mechanism  | Peer-to-peer | Gateway provided by          |
+------------+--------------+------------------------------+
| Conf. tun. | No           | ISP or tunnel broker         |
| AYIYA      | No           | ISP or tunnel broker         |
| GRE        | No           | N/A                          |
| 6a44       | Within domain| ISP                          |
| 6rd        | Within domain| ISP                          |
| 6over4     | Globally     | N/A                          |
| ISATAP     | Within domain| Own organisation             |
| Teredo     | Globally     | Public                       |
| 6to4       | Globally     | Public or ISP                |
| 6bed4      | Globally     | Public or ISP or tunnel broker |
| Auto. tun. | Globally     | N/A                          |
| LISP       | Configurable | ISP or tunnel broker         |
| SEAL       | Configurable | ISP or tunnel broker         |
+------------+--------------+------------------------------+
```

              Topologies Supported per Tunnel Mechanism

6.3. Robustness

   Tunnels may fail for three main reasons: when tunneled packets are
   filtered, typically by a firewall; when a tunnel endpoint IPv4
   address changes; or 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 tunnel 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 tunnel 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 they may have limited performance, depending on distance and usage.

Tunnel mechanisms that use a broadcast or Non-Broadcast Multi-Access (NBMA) communication model may experience failures between some combinations of tunnel endpoints and not others.

The following table lists tunnel mechanisms that provide connectivity to the IPv6 Internet in order of decreasing robustness. (However, even less-robust mechanisms may function well in suitable environments.)

Mechanism	Endpoint address change	Main issues
LISP	automatic	None
6rd	interrupt	None
AYIYA	automatic	Transient NAT mapping issues
Conf. + Heartbeat	interrupt	Proto 41 filtering, competition for NAT mappings (1)
Conf. tun.	failure	Proto 41 filtering, competition for NAT mappings, address change (1)
GRE	failure	Proto 47 filtering, address change
6a44	interrupt	NAT mapping towards peers
6bed4	interrupt	NAT mapping towards peers
6to4	interrupt	Enabled out of the box but filtered, gateway performance (2)
Teredo	interrupt	NAT compatibility, mapping towards peers (3)

Susceptibility of Tunnel 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
      addresses that are not as defined in RFC 1918 [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
      on an incomplete model of NAT.

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 tunnel mechanism that is more robust in all
possible ways than every other tunnel 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 they may need administrative intervention when a tunnel
endpoint address changes.

6.4. Gateway State

There is an additional consideration that is important to operators
of gateways that connect IPv6-in-IPv4 tunnels to the IPv6 Internet:
how much state a tunnel mechanism requires.

6to4, 6rd, 6a44 and 6bed4 require no state at all: when encapsulating
IPv6 packets inside an IPv4 packet, the IPv4 destination address is
directly copied from bits in the IPv6 destination address. This
makes all possible tunneled destinations directly reachable through a
single virtual interface.

Teredo relays maintain a list of peers and are intended to service a
limited number of hosts. The Teredo server, however, is a stateless
gateway component.

With configured tunnels, GRE, AYIYA, and SEAL, there is no direct
mapping from (part of) the IPv6 destination address to the IPv4
destination address. A typical implementation of these mechanisms
has a virtual tunnel interface for each tunnel. Packets are
forwarded to the correct virtual interface through a routing table
lookup. Routing tables can grow very large and remain fast, so the
number of virtual interfaces tends to be the limiting factor for
tunnel gateways. AYIYA and the SixXS Heartbeat Protocol also keep
track of the reachability status of each tunnel.

6.5. Performance

There are several reasons why tunneled connectivity may perform
inferior to native, untunneled connectivity. Inherently, tunnels add
one or more extra headers, and therefore increase overhead. However,
for an Ethernet packet of maximum size (1500 bytes), 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 tunnel encapsulation must then 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 (i.e., increased path stretch can
occur). This may or may not lead to decreased performance.

```
+------------+----------------+---------------------+-------------+
| Mechanism  | Overhead       | Increased path      | Variability |
|            | (bytes)        | stretch             |             |
+------------+----------------+---------------------+-------------+
| Conf. tun. | 20             | may be large        | none        |
| Auto. tun. | 20             | none                | none        |
| 6over4     | 20             | none                | none        |
| GRE        | 28 - 36        | may be large        | none        |
| 6to4       | 20             | may be large        | high        |
| AYIYA      | 72             | may be large        | low         |
| ISATAP     | 20             | none                | none        |
| Teredo     | 28 - 36        | may be large        | high        |
| 6rd        | 20             | small               | low         |
| 6a44       | 20 - 28        | small               | low         |
| 6bed4      | 28             | may be large        | high        |
| LISP       | 36             | small               | low         |
| SEAL       | 24 - variable  | small               | low         |
+------------+----------------+---------------------+-------------+
```

                       Typical Tunnel Performance

7. 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.

   Although, in principle, ingress filtering (BCP 38, [RFC2827]) is
   possible with tunnels, in practice, it is relatively easy for spoofed
   packets to make their way through a tunnel. Not only is it often
   easy to spoof the outer IPv4 header and make false IPv6 packets seem
   to originate from a tunnel broker or gateway, it may also be possible
   for an attacker to route false IPv6 packets through a legitimate
   tunnel broker or gateway. Many tunneling protocols have various
   means of detecting and rejecting such packets, while others have
   limited or no such provisions. For instance, see [RFC3964] for how
   this can be addressed with 6to4.

   So it is important to recognise that unless special measures are
   taken (like [RFC4301]), both IPv4 and IPv6 addresses in tunnel
   packets may be spoofed and cannot be relied upon for access controls.
   Such spoofing was used successfully to discover IPv6-in-IPv4 tunnels
   in [TUNDISC].

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

8. Contributors

Job Snijders contributed text to the points of comparison. Fred
Templin provided the text for SEAL and contributed to the security
considerations. Jeroen Massar, Brian Carpenter, Tina Tsou, John
Mann, Suresh Krishnan, Victor Kuarsingh, Dan Jones, Nejc Skoberne,
and Fred Baker reviewed the document and/or offered suggestions for
improvement.

9. Acknowledgements

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

10. Informative References

   [6BED4]   Van Rein, R., "6bed4: Peer-to-Peer IPv6 on Any
                Internetwork", Work in Progress, July 2013.

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

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

   [GOGO6]   gogo6, "Freenet6: Free and Easy IPv6 Connectivity",
                <http://www.gogo6.com/freenet6>.

   [HEARTBEAT] Massar, J., "SixXS Heartbeat Protocol", Work in Progress,
                June 2005.

   [LISPBETA]   LISP4/LISP6.net, "LISP Beta Network",
                <http://www.lisp4.net/beta-network/>.

   [MAP]   Troan, O., Dec, W., Li, X., Bao, C., Matsushima, S.,
                Murakami, T., and T. Taylor, "Mapping of Address and Port
                with Encapsulation (MAP)", Work in Progress, August 2013.

   [MASSAR]   Massar, J., "AYIYA: Anything In Anything", Work in
                Progress, July 2004.

   [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.

   [RFC2663]   Srisuresh, P. and M. Holdrege, "IP Network Address
               Translator (NAT) Terminology and Considerations",
               RFC 2663, August 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.

   [RFC3964]   Savola, P. and C. Patel, "Security Considerations for
               6to4", RFC 3964, December 2004.

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

   [RFC4301]   Kent, S. and K. Seo, "Security Architecture for the
               Internet Protocol", RFC 4301, December 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.

   [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.

   [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.

   [RFC6724]   Thaler, D., Draves, R., Matsumoto, A., and T. Chown,
               "Default Address Selection for Internet Protocol Version
               6 (IPv6)", RFC 6724, September 2012.

   [RFC6732]   Kuarsingh, V., Lee, Y., and O. Vautrin, "6to4 Provider
               Managed Tunnels", RFC 6732, September 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.

   [SEAL]   Templin, F., "The Subnetwork Encapsulation and Adaptation
               Layer (SEAL)", Work in Progress, October 2013.

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

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

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

   [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/>.

   [TUNDISC]   Colitti, L., Di Battista, G., and M. Patrignani,
               "IPv6-in-IPv4 tunnel discovery: methods and experimental
               results", IEEE eTransactions on Network and Service
               Management (eTNSM), vol. 1, no. 1, p. 2-10, April 2004.

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: the number of
      interactions to set up the tunnel and the 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 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