Teredo: Tunneling IPv6 over UDP through NATs

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

We propose here a service that enables nodes located behind one or several IPv4 NATs to obtain IPv6 connectivity by tunneling packets over UDP; we call this the Teredo service. Running the service requires the help of "Teredo servers" and "Teredo relays"; the Teredo servers are stateless, and only have to manage a small fraction of the traffic between Teredo clients; the Teredo relays act as IPv6 routers between the Teredo service and the "native" IPv6 Internet. A set of Teredo clients, servers and relays constitute a Teredo Network; this document specifies both a global Teredo network, which does not require configuration by the clients, and the possibility to organize local Teredo networks, e.g. within a specific domain or region. A Teredo network is characterized by an IPv6 address prefix and by a unique IPv4 address used by all servers and relays in the network.

1 Introduction

Classic tunneling methods envisaged for IPv6 transition operate by sending IPv6 packets as payload of IPv4 packets; the 6to4 proposal [RFC3056] proposes automatic discovery in this context. A problem with these methods is that they don’t work when the IPv6 candidate node is isolated behind a Network Address Translation device: NAT are typically not programmed to allow the transmission of arbitrary payload types; even when they are, the local address cannot be used in a 6to4 scheme. 6to4 will work with NAT if the NAT and 6to4 router functions are in the same box; we want to cover the relatively frequent case when the NAT cannot be readily upgraded to provide a
6to4 router function.

A possible way to solve the problem is to rely on a set of "tunnel brokers." There are however limits to any solution that is based on such brokers: the quality of service is not very good, since the traffic follows a "dog leg" route from the source to the broker and then the destination; the broker has to provide sufficient transmission capacity to relay all packets and thus suffers a high cost. For these two reasons, we tend to prefer solutions that allow for "automatic tunneling", i.e. let the packets follow a direct path to the destination.

The automatic tunneling requirement is indeed at odds with some of the specificities of NATs. Establishing a direct path supposes that the IPv6 Candidate can retrieve a "globally routable" address that results from the translation of its local address by one or several NATs; it also supposes that we can find a way to bypass the various "per destination protections" that many NATs implement. In this memo, we will explain how IPv6 candidates located behind NATs can enlist the help of "Teredo servers" and "Teredo relays" to learn their "global address" and to obtain connectivity, and how clients, servers and relays can be organized in Teredo networks.

The specification is organized in x sections. Section 2 contains the definition of the terms used in the memo. Section 3 presents the hypotheses on NAT behavior used in the design, as well as the operational requirements that the design should meet. Section 4 presents the models of operation and deployment. Section 6 contains the format of the messages and the specification of the protocol. Section 6 is a discussion of the key design choices. Section 7 contains a security discussion, and section 8 contains IANA considerations.

2 Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This specification uses the following definitions:

2.1 Teredo service

The transmission of IPv6 packets over UDP, as defined in this memo.

2.2 Teredo Client

A node that has some access to the IPv4 Internet and that wants to gain access to the IPv6 Internet. This node may behave either as an IPv6 host or as an IPv6 router.

2.3 Teredo Server

A node that has access to the IPv4 Internet through a globally routable address, and that is used as a helper to provide IPv6 connectivity to Teredo Client. The node is also reachable through the Teredo IPv4 anycast address.
2.4 Teredo Relay
An IPv6 router that can receive traffic destined to Teredo clients and forward it using the Teredo service.

2.5 Teredo IPv6 service prefix
An IPv6 addressing prefix which is used to construct the IPv6 address of Teredo clients.

2.5.1 Global Teredo IPv6 service prefix
An IPv6 addressing prefix whose value is XXXX::/n. (TBD IANA)

2.6 Teredo IPv4 anycast prefix
An IPv4 address prefix from which the Teredo IPv4 anycast address is derived.

2.6.1 Global Teredo IPv4 anycast prefix
An IPv4 address prefix whose value is x.x.x.x/n. (TBD IANA)

2.7 Teredo IPv4 anycast address
An IPv4 address used in the service: the destination address of IPv4 Packets directed to the nearest Teredo server; the source address of IPv4 packets sent by Teredo relays.

2.7.1 Global Teredo IPv4 anycast address
An IPv4 address whose value is x.x.x.y, where the most significant bits match the Global Teredo IPv4 anycast prefix. (TBD IANA)

2.8 Teredo network
A set of clients, servers and relays that are configured to use the same Teredo IPv6 service prefix and the same Teredo IPv4 anycast address.

The Global Teredo Network is composed of the set of Clients, Servers and Relays that use the Global Teredo IPv6 service prefix and the Global Teredo IPv4 anycast address.

2.9 Teredo UDP port
The UDP port number at which Teredo Servers are waiting for packets. This port is also used by Teredo relays to send Teredo packets. The value of this port is PPPP. (TBD IANA)

2.10 Teredo bubble
A packet sent over UDP by a Teredo client in order to trigger a side effect in the NAT.
2.11 Teredo service port

The port through which the Teredo client sends Teredo packets. This port is attached to one of the client’s IPv4 interfaces. The IPv4 address may or may not be globally routable, as the client may be located behind one or several NAT.

2.12 Teredo mapped address and Teredo mapped port

A global IPv4 address and a UDP port that results from the translation by one or several NAT of the IPv4 address and UDP port of a client’s Teredo service port. The client learns these values through the Teredo protocol described in this memo.

2.13 Teredo IPv6 client prefix

A global scope IPv6 prefix composed of the Teredo IPv6 service prefix, the Teredo mapped address and the Teredo mapped port.

2.14 Teredo IPv6 address

A Teredo IPv6 address obtained by combining a Teredo IPv6 client prefix and a node identifier.

2.15 Teredo IPv6 anycast address

A global scope IPv6 address obtained by combining the Teredo IPv6 service prefix, the Teredo IPv6 anycast address, the Teredo UDP port and a null node identifier (i.e., all zeroes).

2.16 Teredo Refresh Interval

The interval during which a Teredo IPv6 Address is expected to remain valid in the absence of "refresh" traffic. For a client located behind a NAT, the interval depends on configuration parameters of the local NAT, or in fact of the combination of NATs on the path to the Teredo server. By default, clients assume an interval value of 30 seconds; a longer value may be determined by local tests, described in section 5.

2.17 Teredo secondary port

A UDP port used to determine the appropriate value of the refresh interval, but not used to carry any Teredo traffic.

2.18 Random link-local address

An IPv6 link-local address chosen at random, as specified in section 5.1.6.

2.19 Teredo IPv4 Discovery Address

An IPv4 multicast address used to discover other Teredo clients in the same IPv4 subnet.

3 Design goals, requirements, and model of operation
The proposed solution transports IPv6 packets as the payload of UDP packets. This is based on the observation that TCP and UDP are the only protocols guaranteed to cross the majority of NAT devices. Relaying packets over TCP would be possible, but would result in a very poor quality of service; relaying over UDP is a better choice.

The design of our solution is based on a set of hypotheses and observations on the behavior of NAT, our desire to provide an "IPv6 provider of last resort", and a list of operational requirements. It results in a model of operation in which the Teredo service is enabled by a set of servers and relays.

3.1 Hypotheses about NAT behavior

NAT devices typically incorporate some support for UDP, in order to enable users in the natted domain to use USP based applications. The NAT will typically allocate a "mapping" when it sees an UDP packet coming through for which there is not yet an existing mapping. The handling of UDP "sessions" by NAT devices differs by two important parameters, the type and the duration of the mappings.

3.1.1 Types of UDP mappings

Experience shows that the implementers of NAT products can adopt widely different treatments of UDP mappings:

1) Some implement the simplest solution, which is to map an internal UDP port, defined by an internal address and a port number on the corresponding host, to an external port, defined by a global address managed by the NAT and a port number valid for that address. In this simple case, the mapping is retained as long as the port is active, and is removed after an inactivity timer. As long as the mapping is retained, any packet received by the NAT for the external port is relayed to the internal address and port. These NATs are usually called "cone NATs".

2) Some implement a more complex solution, in which the NAT not only establishes a mapping for the UDP port, but also maintains a list of external hosts to which traffic has been sent from that port. The packets originating from third party hosts to which the local host has not yet sent traffic are rejected. These NATs are usually called "restricted cone NATs".

3) Instead of keeping just a list of authorized hosts, some NAT implementations keep a list of authorized host and port pairs. UDP packets coming from remote addresses are rejected if the internal host has not yet sent traffic to the outside host and port pair. The NATs are often called "port restricted cone NATs".

4) Finally, some NAT map the same internal address and port pair to different external address and port pairs, depending on the address of the remote host. These NATs are usually called "symmetric NATs".

Measurement campaigns and studies of documentations have shown that most NAT implement either option 1 or option 2, i.e. cone NATs or restricted cone NATs. The Teredo solution ensures connectivity for
all NAT types and all configurations, but it is legitimate to seek
an optimization in the case of cone NAT or restricted cone NATs.

### 3.1.2 Lifetime of UDP mappings

Regardless of their types, UDP mappings are not kept forever. The
typical algorithm is to remove the mapping if no traffic is observed
on the specified port for a "lifetime" period. The Teredo client
that want to maintain a mapping open in the NAT will have to send
some "keep alive" traffic before the lifetime expires. For that, it
needs an estimate of the "lifetime" parameter used in the NAT. We
observed that the implementation of lifetime control can vary in
several ways.

Most NATs implement a "minimum lifetime" which is set as a parameter
of the implementation. Our observations of various boxes showed that
this parameter can vary between about 45 seconds and several
minutes.

In many NATs, mappings can be kept for a duration that exceeds this
minimum, even in the absence of traffic. We suspect that many
implementation perform "garbage collection" of unused mappings on
special events, e.g. when the overall number of mappings exceeds
some limit.

In some cases, e.g. NATs that manage ISDN or dial-up connections,
the mappings will be release when the connection is released, i.e.
when no traffic is observed on the connection for a
period of a few minutes.

Any algorithm used to estimate the lifetime of mapping will have to
be robust against these variations.

---

### 3.2 IPv6 provider of last resort

Teredo is designed to provide an "IPv6 access of last resort" to
nodes that need IPv6 connectivity but cannot use any of the other
transition schemes designed by the NGTRANS working group. This
design objective has several consequences on when to use Teredo, how
to program clients, and what to expect of servers. Another
consequence is that we expect to see a point in time at which the
Teredo technology ceases to be used.

#### 3.2.1 When to use Teredo?

Teredo is designed to robustly enable IPv6 traffic through NAT, and
the price of robustness is a reasonable amount of overhead, due to
UDP encapsulation, transmission of bubbles, and relaying of packets
through the Teredo servers. Nodes that want to connect to the IPv6
Internet SHOULD only use the Teredo service as a "last resort"
option: they SHOULD prefer using direct IPv6 connectivity if it is
locally available or if it is provided by a 6to4 router co-located
with the local NAT, and they SHOULD prefer using the less onerous
"6to4" encapsulation if they can use a global IPv4 address.

#### 3.2.2 Autonomous deployment
In an IPv6 enabled network, the IPv6 service is configured automatically, by using mechanisms such as IPv6 Stateless Address Autoconfiguration [RFC2462] and Neighbor Discovery [RFC2461]. A design objective is to also configure the Teredo service automatically, without requiring any configuration parameter.

3.2.3 Minimal load on servers

During the peak of the transition, there will be a requirement to deploy a large number of servers throughout the Internet. Minimizing the load on the server is a good way to facilitate this deployment. To achieve this goal, servers should be as stateless as possible, and they should also not be required to carry any more traffic than necessary. One way to achieve the load reduction is to enable clients to exchange packets directly whenever possible.

3.2.4 Automatic sunset

Teredo is meant as a short-term solution to the specific problem of providing IPv6 service to nodes located behind a NAT. The problem is expected to be resolved over time by transforming the "IPv4 NAT" into an "IPv6 router". This can be done in one of two ways: upgrading the NAT to provide 6to4 functions, or upgrading the Internet connection used by the NAT to a native IPv6 service, and then adding IPv6 router functionality in the NAT. In either case, the former NAT can present itself as an IPv6 router to the system behind it. These systems will start receiving the "router advertisements"; they will notice that they have IPv6 connectivity, and will stop using Teredo.

3.3 Operational Requirements

3.3.1 Robustness requirement

The Teredo service is designed primarily for robustness: packets are carried over UDP in order to cross as many NAT implementations as possible. The servers are designed to be stateless, which means that they can easily be replicated. We expect indeed to find many such servers replicated at multiple Internet locations.

3.3.2 Protection against denial of service attacks

The Teredo clients obtain mapped addresses and ports from the Teredo servers. The service must be protected against denial of service attacks in which a third party spoofs a Teredo server and sends improper information to the client.

3.3.3 Protection against distributed denial of service attacks

Teredo servers will act as a relay for IPv6 packets. Improperly designed packet relays can be used by denial of service attackers to hide their address, making the attack untraceable. The Teredo service must include adequate protection against such misuse.

3.3.4 Non-requirement of permanent addresses
Our design objective is that any node that has obtained a legitimate IPv4 connection, even behind a NAT, can obtain a globally reachable IPv6 address. We want to achieve this objective in the most automatic manner, without requiring any explicit configuration: explicit configuration is easy to manage by sophisticated users, but experience shows that it can be an insurmountable hurdle for novice users.

The absence of explicit configuration excludes another potential requirement, that nodes obtain a "permanent" address: such permanent assignments require that nodes provide explicit credentials when they connect to the network, which in turn implies explicit management of these credentials.

The absence of permanent addresses can be palliated by "name services" or "rendezvous services" through which the node can publish its address of the moment to its peers; for example, the nodes can be reached through a SIP proxy server, or could use dynamic DNS updates to publish their current address. When permanent addresses are required, they can be obtained and managed by other means; in particular, the Teredo Address can be used as a "care of" address in a Mobile IPv6 service.

### 3.3.5 Compatibility with ingress filtering

Routers may perform Ingress filtering by checking that the source address of the packets received on a given interface is "legitimate", i.e. belongs to network prefixes from which traffic is expected at a network interface. Ingress filtering is a recommended practice, as it thwarts the use of forged source IP addresses by malfeasant hackers, notably to cover their tracks during denial of service attacks.

The simplest form of ingress filtering is perform at the interface between a stub network, e.g. a customer site, and the network through which the stub network accesses the Internet, e.g. a provider network. In such set-up, the routing of packets is symmetric: all packets originating from the customer network are routed through the provider network, and all packets bound to the customer network are routed through the provider network. Routers can perform ingress filtering on interfaces to stub networks by simply performing "reverse path" checks, i.e. checking that packets bound for the source address would have been routed through the interface.

A more complex filtering may be performed at the boundary between two "non stub" networks, e.g. at the peering points between two providers. Contrary to "stub" interfaces, routing at these points is not symmetric. The choice of inter-domain routes is based on a combination of topology and policy considerations; each peer proposes a set of routes that may be reached through the interface, and elects to use a subset of the routes proposed by the other peer. In these conditions, a simple reverse path check would reject a number of perfectly legitimate packets. If a peer really wants to perform ingress filtering, it has to base the check on the complete list of proposed routes rather than the selected subset.
The design of Teredo must be compatible with these two forms of ingress filtering.

4 Model of operation and deployment

A Teredo Network is composed of a set of clients, servers and relays. In this section, we present the model of operation of a given network, and then we present the deployment model.

4.1 Model of operation

The Teredo service requires the cooperation of three kinds of actors: Teredo clients, who want to use IPv6 despite being located behind a NAT, Teredo servers who will facilitate the service, and Teredo relays that provide for the interconnection between the service and the "native IPv6 Internet."

In order to enable the service, the Teredo servers must have IPv6 connectivity and an unencumbered IPv4 connection: they must have a global IPv4 address; and they must have global IPv6 connectivity independently of the Teredo service. These servers must be capable of receiving and sending packets through a specific, topology dependent, IPv4 address, and also through the generic "Teredo anycast IPv4 address."

The Teredo relays must be connected to the IPv6 Internet and must participate in IPv6 routing; they must be able to announce reachability of the "Teredo service IPv6 prefix" over IPv6. They must then be able to relay packets over IPv4 UDP towards Teredo clients. It is very sensible to combine the functions of Teredo server and Teredo relay in a single system.

The primary role of the servers is to enable the NAT traversal. The service is designed in such a way that, when NAT traversal is guaranteed, packets can flow on a direct path between source and destination, without necessarily involving a relay by a Teredo server.

4.1.1 Obtaining an address

The first phase of Teredo operation is the acquisition of a Teredo address prefix by the client. To do this, the client sends a "router solicitation" to the Teredo server, which replies with a router advertisement containing a Teredo prefix.

In order to explain how this works, we will use an hypothetical example: a Teredo client is located at the private IP address 10.0.0.2 in a private network; the NAT connecting this network to the public Internet responds to the local address 10.0.0.1 and is visible as the public address 9.0.0.1.

```
-----------------------
|  Private network      |
| (IPv4) src=9.0.0.1:4096 | NAT | Teredo|
```


When the Teredo client is turned up, it sends an IPv6 router solicitation over UDP to the Teredo IPv4 anycast address, using the Huitema
source address and ports 10.0.0.2:1234. The NAT captures the packet,
establishes a mapping, and changes the source address and port to
9.0.0.1:4096. (These values are indeed just examples.)

The Teredo receives the packet and notes the source address. It uses
it to construct a Teredo IPv6 client prefix:

    PREF:0900:0001:1000::/1

In this prefix, "PREF" is the Teredo IPv6 service prefix of length
1; "0900:0001" is the hexadecimal notation of the IPv4 mapped
address "9.0.0.1"; and "1000" is the hexadecimal notation of the
mapped port "4096".

The router sends the router advertisement back over UDP to the
mapped IPv4 address 9.0.0.1:4096. The IPv4 packet is received by the
NAT, which has presumably remembered the mapping between this
external address and the private address and port pair,
10.0.0.2:1234; the NAT forwards the packet to the Teredo client.

4.1.2    Sending a packet from a Teredo node to a regular IPv6 node

The exchange of packets between a Teredo Node and a regular IPv6
node is presented in the following diagram, in which "A" is a Teredo
client located behind the NAT "N", "S" the Teredo server closest to
"A" in the IPv4 Internet topology, "B" a regular IPv6 node, and "R"
a Teredo Relay close to "B" in the IPv6 Internet Topology.
We assume that A has already established its Teredo address through an RA/RS exchange with S, as explained in the previous paragraph. We will assume that the results of these exchanges are the following:

- A’s "natted" address and port are: 10.0.0.2:1234.
- A’s mapped address and port are: 9.0.0.1:4096.
- A’s IPv6 address has been set to PREF:0900:0001:1000::A
- The Teredo IPv4 anycast address and port are: t.t.t.t:p
- B’s IPv6 address is: 2000::B

The IPv6 packet sent by A to B carries the source address PREF:0900:0001:1000::A (A’s address), and the destination address 2000::B (B’s address). The packet is encapsulated by A in a UDP datagram, from source address and port 10.0.0.2:1234, to source address and port t.t.t.t:p.

The packet is received by the NAT N. N uses the existing mapping for 10.0.0.2:1234, and replaces the UDP source address and port by the mapped values 9.0.0.1:4096, before forwarding the packet.

The packet is received over IPv4 by S, the closest server. S discards the IPv4 and UDP header, and forwards the content of the packet over IPv6, which will be received by B, and which will appear to come from A’s address, PREF:0900:0001:1000::A.
4.1.3 Sending a packet from a regular IPv6 node to a Teredo node

The following diagram uses the same notations as those in the previous section to present the exchange from the IPv6 Internet host "B" to the Teredo client "A".

--- IPv6 Internet ---

IPv6 Internet
    +-----+
    | B    |
    +-----+

    :     
    v

    +-----+
    :     
    v

--- IPv4 Internet ---

IPv4 Internet
    +-----+
    :     

    :     
    v

--- Natted domain ---

A

The IPv6 packet sent by B to A carries the source address 2000::B (B’s address), and the destination address PREF:0900:0001:1000::A (A’s address). The packet is sent over IPv6, and is routed towards the closest Teredo relay, i.e. the closest node that advertises the prefix PREF::/l.

The relay R receives the device, and extracts the IPv4 address and UDP port number t.t.t.t:4096 from the destination address PREF:0900:0001:1000::A. The IPv6 packet is encapsulated by R in a IPv4/UDP datagram, from source address and port t.t.t.t:p, to source address and port 9.0.0.1:4096.

The packet is routed over the IPv4 Internet and received by the NAT N. Whatever the type of the NAT N, cone or symmetric, the NAT has an existing UDP mapping in place, since A already sent packets to the address t.t.t.t:p. The NAT N uses this mapping and replaces the external destination address and port by A’s address and port, 10.0.0.2:1234.

The packet is then forwarded in the natted domain, and received by...
A. A discards the IPv4 and UDP header, and processes the IPv6 packet.

We note that a key requirement for this exchange to work is the possibility for the router R to source packets from the IPv4 address and port used by S. Only routers that meet this requirement should advertise reachability of the Teredo prefix PREF::/1 in the IPv6 routing infrastructure.

4.1.4 Exchanges between two Teredo nodes

The following diagram shows two Teredo clients, A and B, connected through the NATs N1 and N2. The exchanges will use the Teredo server S1, close to A, and S2, close to B. We will assume that the NAT N2 is a "restricted cone" or "port restricted cone" NAT; the only assumption about the NAT N1 is that it is not a "symmetric" NAT.

Both S1 and S2 can serve the Teredo address and port, t.t.t.t:p. We will assume that A and B have already obtained Teredo addresses by RS/RA exchanges with S1 and S2. In the example, we will assume the following values:

- A’s "natted" address and port are: 10.0.0.2:1234.
- A’s mapped address and port are: 9.0.0.1:4096.
- A’s IPv6 address has been set to PREF:0900:0001:1000::A
- B’s "natted" address and port are: 10.0.0.3:3456.
- B’s mapped address and port are: 8.0.0.1:1024.
- B’s IPv6 address has been set to PREF:0800:0001:0400::B
- The Teredo IPv4 anycast address and port are: t.t.t.t:p

A has learned B’s address, for example from the DNS, and starts an UDP or TCP exchange with B. The first data packet will be sent from A’s IPv6 address (PREF:0900:0001:1000::A) to B’s IPv6 address (PREF:0800:0001:0400::B). Since A has no prior knowledge of B, the first packet will have to be sent through the Teredo server. The packet is encapsulated in an IPv4/UDP datagram, with source address and port 10.0.0.2:1234, destination address and port t.t.t.t:p.

The first packet passes through N1, which replaces the source address and port by the mapped values 9.0.0.1:4096. The packet arrives then to S1, which discards the IPv4 and UDP header, and extract’s B’s mapped address from the destination IPv6 address. The IPv6 packet is then encapsulated in a new IPv4/UDP datagram, with source address and port 8.0.0.1:1024. This packet is received by N2; as a mapping already exists for this address pair, the destination address and pair are rewritten to 10.0.0.3:3456, and the packet is received by B.

At the same time it sends the packet to B through the Teredo server, A will also create a bubble that it will send to B on a direct path. The bubble is a UDP packet sent from A’s address (10.0.0.2:1234) to B’s mapped address and port (8.0.0.1:1024). The bubble is sent through the NAT N1, where it creates a mapping, and then forwarded to the NAT N2, where it is probably discarded, since there is no established mapping between A’s address (9.0.0.1:4096) and B.

When B respond to A, the data packet is also sent through the Teredo server S2, much like the first data packet. B, however, will also send a bubble to A. The bubble is from B’s address (10.0.0.3:3456) to A’s mapped address and port (9.0.0.1:4096). The bubble is sent through the NAT N2, where it creates a mapping, and then forwarded to the NAT N1. As the first bubble established a mapping between A’s address (9.0.0.1:4096) and B, the bubble is accepted and the makes it way to A.

At this point, A knows that it has received a direct packet from B. A’s next packet will be sent on the direct path, and will make it through N1 since a mapping has already been establish. Similarly, as soon as B receives a packet from A, B’s packets will be sent on the direct path.

4.2 Deployment model

The model of operation makes a critical assumption, that all servers and relays that cooperate in a Teredo Service will use the Teredo IPv4 anycast address as the source address of the packets that they will send to Teredo clients. The use of a single address is necessary, because we can only assume that a single UDP mapping will be actively maintained by the Teredo clients. However, this use of
"anycast" is unusual and may interfere with "ingress filtering". As a consequence, we must support two deployment models: the global Teredo network and the specific Teredo networks.

4.2.1 Deployment in the Global Teredo Network

The global Teredo network is composed of clients, relays and servers that use the "Global Teredo IPv6 prefix" and the "Global Teredo IPv4 anycast address."

Network administrators who choose to deploy relays and servers as part of the Global Teredo Network MUST ensure that this deployment will be compatible with "ingress filtering". Global Teredo Network servers and relays SHOULD only be deployed in a stub network if the interface to the provider network is explicitly configured to allow packets sent from the Global Teredo IPv4 anycast address; Global Teredo Network servers and relays SHOULD only be deployed in an autonomous system if the autonomous system is ready to announce a route to the Global Teredo IPv4 anycast prefix to all its peers.

Ingress filtering requirement MUST also be met at the IPv6 layer. Global Teredo Network servers and relays should only be deployed in an autonomous system if the autonomous system is ready to announce a route to the Global Teredo IPv6 prefix to all its peers.

A network operator who is not ready to announce the service to the whole Internet SHOULD NOT operate servers and relays for the Global Teredo Network.

4.2.2 Deployment of a specific Teredo network

A specific Teredo network is composed of clients, relays and servers that use a "Teredo IPv6 prefix" and the "Teredo IPv4 anycast address" other than the "Global Teredo IPv6 prefix" and the "Global Teredo IPv4 anycast address."

Network managers who want to deploy a specific Teredo network SHOULD select an IPv6 prefix in the IPv6 address space managed by the local network, and an IPv4 address in the IPv4 address associated to this network. This IPv6 prefix and IPv4 address become the identifying parameters of a "specific Teredo network", provided by servers and relays that are all located within the network managed by the network operator. The local clients will have to be configured to use these parameters.

5 Specification of clients, servers and relays

The Teredo service is realized by having clients interact with Teredo servers through the Teredo service protocol. The clients will also receive IPv6 packets through Teredo relays.

The Teredo server is designed to be stateless. It waits for Teredo
requests and for IPv6 packets on the Teredo UDP port; it processes
the requests by sending a response to the appropriate address and
port; it forwards Teredo IPv6 packets to the appropriate IPv4
address and UDP port.

The Teredo relay advertises reachability of the Teredo service
prefix over IPv6. It forwards Teredo IPv6 packets to the appropriate
IPv4 address and UDP port.

5.1 Message formats

5.1.1 Teredo IPv6 addresses

Teredo IPv6 addresses are composed of four components:

+--------------------+------------------+
<table>
<thead>
<tr>
<th>Prefix</th>
<th>IPv4 address</th>
<th>Port</th>
<th>Node identifier</th>
</tr>
</thead>
</table>
+--------------------+------------------+

The n bit prefix is the "Teredo IPv6 Prefix."

The next 32 bits contain a globally routable IPv4 address, i.e. the
Teredo mapped address.

The next 16 bits contain a port number associated with that address,
i.e. the Teredo mapped port.

The last (80 - n) bits contain a node identifier.

5.1.2 Teredo IPv6 packets encapsulation

Teredo IPv6 packets are transmitted as UDP packets [RFC768] within
IPv4 [RFC791]. The source and destination IP addresses and UDP port
take values that are specified in this section.

5.1.3 Maximum Transmission Unit

Since Teredo uses UDP as an underlying transport, a Teredo
Maximum Transfer Unit (MTU) could potentially be as large as the
payload of the largest valid UDP datagram (65507 bytes). However,
since Teredo packets can travel unto unpredictable path over the
Internet, it is best to contain this MTU to a small size, in order
to minimize the effect of packet fragmentation and reassembly. The
default link MTU assumed by a host, and the link MTU supplied by a
Teredo server during Router Advertisement SHOULD normally be set to
the minimum IPv6 MTU size of 1280 bytes [RFC2460].

Teredo implementations SHOULD NOT set the "do not fragment" (DF) bit
of the encapsulating IPv4 header.

5.1.4 Teredo bubble
A Teredo bubble is a minimal UDP packet sent to a specific destination to "prime" a NAT for later accepting packets from this destination. The bubble, as formatted by the sender, has the following parameters:

- IPv4 source address: the IPv4 address of the sender
- IPv4 destination address: the Teredo mapped address of the target
- UDP source port: the Teredo service port of the sender
- UDP destination port: the Teredo mapped port of the target
- UDP payload: a minimal IPv6 packet, as follows
  - IPv6 source: the Teredo IPv6 address of the sender
  - IPv6 destination: the Teredo IPv6 address of the target
  - IPv6 payload type: 59 (No Next Header, as per [RFC2460])
  - IPv6 payload length: 0
  - IPv6 hop limit: 1

When the bubble reaches its destination, the sender’s NAT will have rewritten the IPv4 source address and the UDP source port to the Teredo map address and Teredo mapped port of the sender; the receiver’s NAT will have rewritten the IPv4 destination address and UDP destination port to the IPv4 address and Teredo service port of the receiver.

5.1.5 Server link local address

The Teredo server uses a link local address as the IPv6 source address of the router advertisement messages used in the qualification and maintenance procedure. The server link local address is an IPv6 source address composed of the 80 bit link local prefix: FE80::/80, and of a 48 bit node identifier composed of a 16 bit port number (in network order) and a 32 bit IP address (in network order):

+---------------------------+------------------+
| FE80 | 64 zero bits | Port | IPv4 (32 bits) |
+---------------------------+------------------+

The IPv4 address is a specific unicast address of an interface through at which the server is ready to receive Teredo packets, at the specified port. This specific address and port pair can be used by trouble shooting procedures, e.g. to check that a specific Teredo server is still operational and reachable.
5.1.6 Random link local address

The Teredo client uses a random link local address as the IPv6 source address of the router solicitation messages used in the qualification and maintenance procedure. The random link local address is an IPv6 source address composed of the 64 bit link local prefix: FE80::/64, and of a 64 bit node identifier picked at random.

Implementers SHOULD ensure that at least one of the most significant 16 bits of the node identifier is not null in order to avoid confusion with the Server link local address.

5.1.7 Teredo local discovery bubbles

The Teredo local discovery bubbles are variation of the Teredo bubbles used by the optional local discovery procedure. The discovery bubble has the following format:

- IPv4 source address: the IPv4 address of the sender
- IPv4 destination address: the Teredo IPv4 Discovery Address
- IPv4 ttl: 1
- UDP source port: the Teredo service port of the sender
- UDP destination port: the Teredo UDP port
- UDP payload: a minimal IPv6 packet, as follows
  - IPv6 source: the Teredo IPv6 address of the sender
  - IPv6 destination: the all nodes on link multicast address
  - IPv6 payload type: 59 (No Next Header, as per [RFC2460])
  - IPv6 payload length: 0
  - IPv6 hop limit: 1

5.2 Teredo Client specification

A Teredo client expects to exchange IPv6 packets through an UDP port, the Teredo service port. The client will maintain the following variables that reflect the state of the Teredo service:

- Teredo connectivity status,
- Mapped address and port number associated with the Teredo service port,
Before sending any packets, the client must perform the Teredo qualification procedure, which determines the Teredo connectivity status, the mapped address and port number, and the Teredo IPv6 prefix. If the qualification is successful, the client may use the Teredo service port to transmit and receive IPv6 packets, according to the transmission and reception procedures; these procedures use the "list of recent peers". For each peer, the list contains:

- The mapped IPv4 address and mapped UDP port of the peer,
- If local discovery is used, the local address and local port of the peer,
- The date and time of the last reception from the peer,
- The date and time of the last transmission to the peer,
- The number of bubbles transmitted to the peer.

The list of peers is used to "optimize" the transmission of IPv6 packets by using a "direct path" for the recently active peers. The list of peers could grow over time. Clients should implement a list management strategy, such as for example deleting the least recently used entries. Clients should make sure that the list has a sufficient size, so that most traffic goes over the "direct path."

The client must regularly perform the maintenance procedure in order to guarantee that the Teredo service port remains usable; the need to use this procedure or not depends on the delay since the last interaction with the Teredo server. The refresh procedure takes as a parameter the "Teredo refresh interval". This parameter is initially set to 30 seconds; it can be updated as a result of the optional "interval determination procedure." The randomized refresh interval is set to a value randomly chosen between 75% and 100% of the refresh interval.

In order to avoid triangle routing for stations that are located behind the same NAT, the Teredo clients MAY use the optional local client discovery procedure defined in section 5.2.7.

5.2.1 Qualification procedure

The purpose of the qualification procedure is to establish the status of the local IPv4 connection, and to determine the Teredo IPv6 client prefix of the local Teredo interface. The procedure starts when the service is in the "initial" state, and results in a "qualified" state if successful, in an "off-line" state if unsuccessful.
Initially, the Teredo connectivity status is set to "Initial". At this point, the client will choose a random link local address, as defined in 5.1.6.

When the interface is initialized, the system first performs the "start action" by sending a Router Solicitation Message, as defined in [RFC2461]. The client picks a random link local address and uses it as the IPv6 source of the message; the IPv6 destination of the RS is the all-routers multicast address; the packet will be sent over UDP from the service port to the Teredo anycast address and Teredo UDP port. The connectivity status moves then to "Starting".

In the starting state, the client waits for a router advertisement from the Teredo server. If no response comes within a time-out T, the client should repeat the start action, by resending the Router Solicit packet. If no response has arrived after N repetitions, the client concludes that it cannot use UDP, and that the Teredo service is not available; the status is set to "Off-line." In accordance with [RFC2461], the default time-out value is set to T=4s, and the maximum number of repetitions is set to N=3.

If a response arrives, the client checks that the response is a valid router advertisement as defined in [RFC2461], that the IPv6 destination address is equal to the random link local address used in the router solicitation, and that it contains exactly one advertised Prefix Information parameter. This prefix should be a valid Teredo IPv6 client prefix: the first bits should contain the Teredo service prefix. If this is the case, the client learns the Teredo mapped address and Teredo mapped port associated to the local Teredo service port:

- the 32 bits after the prefix are the client’s Teredo mapped
address,

- the next 16 bits are the client’s Teredo mapped port.

The source address of the Router Advertisement is the link local server address of the Teredo server. This address must be built using an IPv4 unicast address of the server, and a port number at which the server is waiting for packets, as specified in section 5.1.5.

If the response is a valid router advertisement, from a valid Teredo source address, with a matching destination address, and containing a valid Teredo address prefix, the client should build a Teredo IPv6 address using the Teredo IPv6 client prefix learned from the RA and locally selected identifier. The client is qualified, and can start using the Teredo service.

5.2.2 Packet reception

The Teredo client receives packets over the Teredo interface. The role of the packet reception procedure, besides receiving packets, is to maintain the date and time of the last interaction with the Teredo server, and the "list of recent peers." Each entry in the list contains an IPv4 address, a UDP port, and a date and time.

When a UDP packet is received over the Teredo service port, the Teredo client checks that it contains a valid IPv6 packet as specified in [RFC2460]; if this is not the case, the packet is silently discarded. Then, the Teredo client examines the IPv4 source address and port number from which the packet is received. If these values match the Teredo IPv4 anycast address and Teredo port, the client updates the "date and time of the last interaction with the Teredo server" to the current date and time.

If the values are different, the client examines the IPv6 source address of the encapsulated packet. If the IPv6 source address is not a valid Teredo IPv6 address, or if the IPv4 address embedded in the Teredo address is not in the format of a global unicast address, the packet MUST be silently discarded. In the other cases, the client performs address check validation.

If the source IPv6 address is a Teredo IPv6 address, the client checks extracts the "peer IPv4 address" and "peer UDP port" corresponding to that address. If these addresses and port match the source IP address and source UDP port of the packet, the client has received a direct packet from a peer; if the address don’t match and the client implements the local client discovery procedure described in section 5.2.7, the client checks whether there is an entry corresponding to the peer address and peer port in the list of recent peers; if the source address and source port match the local address and local port of the peer, the client has received a direct packet from a local peer. In the other cases the packet SHOULD be
silently discarded unless it is a "local discovery bubble", in which case the client MAY perform the processing described in section 5.2.7.

If the packet is accepted and the source IPv6 address is a Teredo IPv6 address, the client check whether the address and port pair are already represented in the "list of recent peers". If this is the case, the client MUST update the date and time of last reception from this peer. If the pair is not yet represented, the client MUST create a new list entry for the pair, setting the last reception date to the current date and time, the last transmission date to 30 seconds before the current date, and the number of bubbles to zero; the local address and local port are set to null values, indicating a non-local peer.

Whatever the IPv4 source address and UDP source port, the client that receives an IPv6 packet from a Teredo IPv6 source address MAY send a Teredo Bubble towards that target, as specified in section 5.2.5.

5.2.3 Packet transmission

When a Teredo client has to transmit a packet over a Teredo interface, it examines the destination IPv6 address. If the destination is not a Teredo IPv6 address, the packet is sent over UDP to the Teredo IPv4 anycast address and Teredo UDP port.

If the destination is a Teredo IPv6 address, the client extracts the "peer IPv4 address" and "peer UDP port" corresponding to that address. Before sending the packet, the Teredo Client MUST check that the IPv4 destination address is in the format of a global unicast address; if this is not the case, the packet MUST be silently discarded.

If there is an entry for this address and port pair in the list of recent Teredo peers, the client checks whether the entry is still valid: an entry associated with a local peer is valid if last reception date and time associated with that list entry is less that 600 seconds from the current time; an entry associated with a non-local peer is valid if the last reception date and time associated with that list entry is less that 30 seconds from the current time. (Local peer entries can only be present if the client uses the local discovery procedure discussed in section 5.2.7.)

If there is a valid entry associated to a local peer, the packet SHOULD be sent to a local IPv4 address and local port documented in the peer entry; if there is valid peer entry associated to a non-local peer, the packet SHOULD be sent over UDP directly to the specified "peer IPv4 address" and "peer UDP port". In both cases, the transmission date for that list entry should be set to the current time.
If there is no entry in the list, or if the date and time associated to the entry is too old, the client SHOULD send the IPv6 packet over IPv4 UDP to the Teredo IPv4 anycast address and Teredo UDP port; the client MAY send a Teredo Bubble towards the destination IPv6 address, as specified in section 5.2.5.

5.2.4 Maintenance

The Teredo client must ensure that the mappings that it uses remain valid. It does so by checking that packets are regularly received from the Teredo server.

At regular intervals, the client MUST check the "date and time of the last interaction with the Teredo server", to ensure that at least one packet has been received in the last Randomized Teredo Refresh Interval. If this is not the case, the client SHOULD select a new randomized link local address and use this address to send a router solicitation message to the Teredo IPv6 Anycast address.

When the router advertisement is received, the client SHOULD check its validity as specified in 5.2.1; invalid advertisements are silently discarded. If the advertisement is valid, the client MUST check that the advertised prefix corresponds to the current Teredo service prefix. If this is not the case, the mapping has changed; the client MUST check that the new prefix is valid, as specified in the qualification procedure. It should then invalidate the addresses derived from the old prefix.

After receiving a router advertisement, the Randomized Teredo Refresh Interval is reset to a new value, randomly chosen between 75% and 100% of the Teredo Refresh Interval.

5.2.5 Sending Teredo Bubbles

The Teredo client may decide to send a bubble towards another Teredo client, either after a packet reception or after a transmission attempt, as explained in sections 5.2.2 and 5.2.3.

When a Teredo client attempts to send a bubble, it extracts the mapped IPv4 address and mapped UDP port from the Teredo IPv6 address of the target. It then checks whether there is already an entry for this address and port pair in the current list. If the pair is not yet represented, the client MUST create a new list entry for the pair, setting the last reception date and the last transmission date to 30 seconds before the current date, and the number of bubbles to zero.

Bubbles may be lost in transit, and it reasonable to enhance the reliability of the Teredo service by allowing multiple transmissions; however, bubbles will also be lost systematically in certain NAT configurations. In order to strike a balance between reliability and unnecessary retransmissions, we specify the
following:

- If the client which implements the local discovery procedure it SHOULD NOT send a bubble to a local peer;

- The client MUST NOT send a bubble if the last transmission date and time is less than 10 seconds before the current date and time;

- The client MUST NOT send a bubble if it has already sent 4 bubbles to the peer in the last 300 seconds without receiving a direct response.

In the other cases, the client MAY proceed with the transmission of the bubble, which MUST be composed as specified in section 5.1.4. When transmitting the bubble, the client MUST update the last transmission date and time to that peer, and must also increment the number of transmitted bubbles.

5.2.6 Optional Refresh Interval Determination Procedure

In addition to the regular client resources described in the beginning of this section, the refresh interval determination procedure uses an additional UDP port, the Teredo secondary port, and the following variables:

- Teredo secondary connectivity status,
- Mapped address and port number of the Teredo secondary port,
- Teredo secondary IPv6 prefix associated with the secondary port,
- Teredo secondary IPv6 address derived from this prefix,
- Date and time of the last interaction on the secondary port,
- Maximum Teredo Refresh Interval.
- Candidate Teredo Refresh Interval.

The secondary connectivity status, mapped address and prefix are determined by running the qualification procedure on the secondary port. When the client uses the interval determination procedure, the qualification procedure MUST be run for the secondary port immediately after running it on the service port. If the secondary qualification fails, the interval determination procedure will not be used, and the interval value will remain to the default value, 30 seconds. If the secondary qualification succeeds, the maximum refresh interval is set to 120 seconds, and the candidate Teredo refresh interval is set to 60 seconds, i.e. twice the Teredo refresh interval. The procedure is then performed at regular intervals, until it concludes:

1) wait until the candidate refresh interval is elapsed after the last interaction on the secondary port;

2) send a Teredo Bubble to the Teredo secondary IPv6 address, through the service port.
3) wait for reception of the bubble on the secondary port. If a timer of 2 seconds elapses without reception, repeat step 2 at most three times. If there is still no reception, the candidate has failed; if there is a reception, the candidate has succeeded.

4) if the candidate has succeeded, set the Teredo refresh interval to the candidate value, and set a new candidate value to the minimum of twice the new refresh interval, or the average of the refresh interval and the maximum refresh interval.

5) if the candidate has failed, set the maximum refresh interval to the candidate value. If the current refresh interval is larger than or equal to 75% of the maximum, the determination procedure has concluded; otherwise, set a new candidate value to the average of the refresh interval and the maximum refresh interval.

6) if the procedure has not concluded, perform the maintenance procedure on the secondary port, which will reset the date and time of the last interaction on the secondary port, and may result in the allocation of a new Teredo secondary IPv6 address; this would not affect the values of the refresh interval, candidate interval or maximum refresh interval.

The secondary port MUST NOT be used for any other purpose than the interval determination procedure. If a spurious packet is received on the secondary port, the client SHOULD repeat the maintenance procedure on this port and reset the date and time of the last interaction on the secondary port.

5.2.7 Optional local client discovery procedure

It is desirable to enable direct communication between Teredo clients that are located behind the same NAT, without forcing a systematic relay through a Teredo server. It is hard to design a general solution to this problem, but we can design a partial solution when the Teredo clients are connected to the same link.

A Teredo client who wishes to enable local discovery SHOULD wait for discovery bubbles to be received on the Teredo IPv4 Discovery Address, and should send a local discovery bubbles to the Teredo IPv4 Discovery Address at random intervals, uniformly distributed between 200 and 300 seconds.

The local discovery procedure carries a denial of service risk, as malevolent nodes could send fake bubbles to unsuspecting parties, and thus capture the traffic originating from these parties. The risk is mitigated by two factors:

- The Teredo client is located behind a NAT which will...
- The Teredo IPv4 Discovery Address is a "link only" multicast address, which implies that packets sent to this address, will not be forwarded across routers.

To benefit from the "link only multicast" protection, the clients should silently discard all local discovery bubbles that are received over a unicast address. To further mitigate the denial of service risk, the client MUST silently discard all local discovery bubbles whose IPv6 source address is not a well formed Teredo IPv6 address, or whose IPv4 source address does not belong to the local IPv4 subnet; the client MAY decide to silently discard all local discovery bubbles whose Teredo IPv6 address does not include the same mapped IPv4 address as its own.

If the bubble is accepted, the client checks whether there is an entry in the list of recent peers that correspond to the mapped IPv4 address and mapped UDP port associated with the source IPv6 address of the bubble. If there is such an entry, the client MUST update the local peer address and local peer port parameters to reflect the IPv4 source address and UDP source port of the bubble. If there is no entry, the client MUST create one, setting the local peer address and local peer port parameters to reflect the IPv4 source address and UDP source port of the bubble the last reception date to the current date and time, the last transmission date to 30 seconds before the current date, and the number of bubbles to zero.

5.3 Teredo Server specification

The Teredo server is designed to be stateless. The Teredo server waits for incoming UDP packets at the Teredo Relay Port. The incoming packets may be sent to either the Teredo IPv4 Anycast Address, or to the specific unicast address of the Teredo Server.

The Teredo server acts as an IPv6 router. As such, it will receive Router Solicitation messages, to which it will respond with router advertisement packets as explained in the "router solicitation" procedure; it may also receive other packets, for example ICMP packets, which are processed according to the IPv6 specification.

The Teredo service may be made available to neighboring Autonomous Systems by advertising the Teredo IPv4 Anycast prefix in the IPv4 EGP, as specified in 5.3.4. The domain managers must be ready to perform fault isolation as specified in 5.3.5.

5.3.1 Processing of Teredo IPv6 packets

Upon reception of a packet on the Teredo port, the Teredo server will first check that the UDP payload contains a valid IPv6 packet; if this is not the case, the packet will be silently discarded.

Before processing the packet, the Teredo server MUST check the validity of the encapsulated IPv6 source address, the IPv4 source address...
The IPv4 source address and the UDP source port:

1) If the IPv4 source address does not belong to a range of IPv4 addresses for which the Teredo server is providing service, the server MAY silently discard the packet.

2) If the IPv4 source address is not in the format of a global unicast address, the packet MUST be silently discarded.

3) If the UDP content is not a well formed IPv6 packet, the packet MUST be silently discarded.

4) If the IPv6 source address is an IPv6 link local address, the IPv6 destination address is the "all routers on link" multicast address, and the packet contains an ICMPv6 "router solicitation", the packet MUST be accepted.

5) If the IPv6 source address is a Teredo IPv6 address, and if the IPv4 address and UDP port embedded in that address match the IPv4 source address and UDP source port, the packet MUST be accepted.

6) In all other cases, the packet MUST be silently discarded.

The Teredo server will then check the IPv6 destination address of the encapsulated IPv6 packet.

If the IPv6 destination address is either the Teredo IPv6 anycast address or the Teredo IPv6 address associated to the local interface, the Teredo server processes the packet; it may in particular have to process "router solicitation" messages.

If the IPv6 destination address is a valid Teredo IPv6 address, the Teredo Server MUST check that the IPv4 address derived from this IPv6 address is in the format of a global unicast address; if this is not the case, the packet MUST be silently discarded. If the address is valid, the Teredo server encapsulates the IPv6 packet in a new UDP datagram, in which the following parameters are set:

- The destination IPv4 address is derived from the IPv6 destination.
- The source IPv4 address is the Teredo IPv4 anycast address.
- The destination UDP port is derived from the IPv6 destination.
- The source UDP port is set to the Teredo UDP Port.

If the destination IPv6 address is not a global scope IPv6 address, the packet MUST be silently discarded. In the other cases, if the destination address is not a Teredo IPv6 address, the packet should be relayed to the IPv6 Internet using regular IPv6 routing.

5.3.2 Processing of router solicitations
When the Teredo server receives a router solicitation message (RS, [RFC2641]), it retains the IPv4 address and UDP port from which the solicitation was received; these become the Teredo mapped address and Teredo mapped port of the client. The router uses these values to compose a Teredo IPv6 Prefix.

The Teredo server responds to the router solicitation by sending a Router Advertisement [RFC2641]. The router advertisement MUST advertise the Teredo IPv6 prefix composed from the mapped address and mapped port. The IPv6 source address must be set to the Teredo link local server address associated to the local interface. The IPv6 destination address is set to the IPv6 source address of the RS. The Router Advertisement must be sent over UDP to the Teredo mapped address and Teredo mapped port of the client; the IPv4 source address and UDP source port should be set to the Teredo IPv4 anycast address and Teredo Port. Before sending the packet, the Teredo relay MUST check that the IPv4 destination address is in the format of a global unicast address; if this is not the case, the packet MUST be silently discarded.

5.3.3 Processing of bubbles

Teredo servers may receive Teredo Bubbles; these messages are silently discarded.

5.3.4 Advertising of the Teredo IPv4 anycast prefix

If the managers of an IPv4 autonomous domain that includes Teredo servers want to provide the Teredo service to neighbor Autonomous Systems, they MAY advertise reachability of the Teredo IPv4 anycast prefix using the EGP of their IPv4 autonomous system, as if it were a connection to an external network. When this advertisement is done using BGP, the initial AS path MUST contain the AS number of the announcing AS. The AS path should also include an indication of the actual router providing the service; there is a suggestion to perform this function by documenting the specific IPv4 address of the Teredo server in the BGP aggregator attribute of the path; further work is needed on this point.

Any Teredo server corresponding to this specification SHOULD include a monitoring function, to check that the Teredo service is operational. The route leading to the Teredo IPv4 anycast prefix MUST NOT be announced if the service is not operational.

5.3.5 Fault isolation

When an error is reported, e.g., by a user, the domain manager should be able to find the specific Teredo Server that is causing the problem.

The first step of fault isolation is to retrieve the specific IPv4
address of the server used by the user. If the server is located within the domain, this information will have to be retrieved from the IGP tables. If the service is obtained through a peering agreement with another domain, the information will be retrieved from the EGP data, e.g., the BGP path attributes.

The second step is obviously to perform connectivity tests using the specific IPv4 address.

5.4 Teredo Relay specification

Teredo relays are IPv6 routers that advertise reachability of the Teredo service IPv6 prefix through the IPv6 routing protocols. Teredo relays will receive IPv6 packets bound to Teredo clients. Teredo relays MUST be able to use the Teredo IPv4 anycast address as the source address of UDP packets.

By definition, the Teredo relays will only process IPv6 packets whose IPv6 destination address is a valid Teredo IPv6 address. Before processing these packets, the Teredo Server MUST check that the IPv4 destination address embedded in the Teredo IPv6 address is in the format of a global unicast address; if this is not the case, the packet MUST be silently discarded. If the address is valid, the Teredo server encapsulates the IPv6 packet in a new UDP datagram, in which the following parameters are set:

- The destination IPv4 address is derived from the IPv6 destination.
- The source IPv4 address is the Teredo IPv4 anycast address.
- The destination UDP port is derived from the IPv6 destination.
- The source UDP port is set to the Teredo UDP Port.

It is obviously desirable to combine the functions of Teredo relay and Teredo server, but this is not mandatory.

5.4.1 Difference between Teredo Relays and Teredo Servers

The requirement on relays is that they be able to send packets from the Teredo IPv4 Anycast address, and that they advertise reachability of the Teredo IPv6 service prefix. The requirement on servers is that they have an unencumbered IPv4 connectivity, that they can receive packet sent to the Teredo IPv4 Anycast address, that they process these packets according to the Teredo server specification, that they can send packets from the Teredo IPv4 Anycast address, and that they can forward packets to IPv6.

The additional requirement for a Teredo server to become a Teredo relay is: advertise the Teredo IPv6 service prefix in the IPv6 routing fabric. This may or may not be easy to achieve; it could be hard in some cases, e.g. if the Teredo server’s connectivity is by
means of the 6to4 service. In any case, if a Teredo server receives IPv6 packets bound to a Teredo destination, it MUST process them according to the Teredo relay specification.

The additional requirement for a Teredo relay to become a Teredo server is: to advertise reachability of the Teredo IPv4 Anycast address in the IPv4 routing fabric; to process packets received on this address according to the Teredo server specification. A relay MUST NOT advertise IPv4 reachability of the Teredo IPv4 Anycast address if it cannot process the packets according to the Teredo server specification. In general, IPv4 nodes that cannot behave as Teredo Servers MUST NOT advertise reachability of the Teredo IPv4 anycast address, or of any other address derived from the Teredo IPv4 anycast prefix.

Many routing configurations implement IPv4 ingress filtering based on "reverse path forwarding" information, i.e. would only accept a packet with a given source address from a specific if the interface is a valid path for delivering packets to that address. When this is the case, it is not possible to implement the Teredo relay function without also implementing the Teredo server function.

6 Discussion of the solution

This section is an attempt at answering various questions about the design choices.

6.1 Why do we have bubbles and lists of peers?

The Teredo procedures are designed to enable direct transmission between clients whenever possible. The role of the bubble is to enable traversal of "restricted cone" and "port restricted cone" NATs; direct transmission to a "cone" NAT is almost always possible (apart from the situation shown in 5.2); direct transmission to a symmetric NAT is generally not possible. The reception of a bubble is a proof that a packet did cross all the NATs on the way, and thus that the destination is reachable. The reachable destinations are listed in the "list of peers"; we know that we can safely use direct path transmission to these destinations. All other destinations will be reached through the Teredo server.

6.2 Could we make do with fewer bubbles?

It is true that if we knew the type of the NAT at each end, we could diminish the amount of bubbles that we send. For example, a Teredo client behind a cone NAT never needs to send or receive bubbles: the NAT will let packet through from any destination. Similarly, a Teredo client behind a symmetric NAT should not bother sending bubbles, since these bubbles will never have any effect on the NAT. On the other end, bubbles are useful for restricted NATs, which according to our sample represent about half the implementations.
A problem with any "smart" approach is that it requires a "characterization" procedure for the local NAT, which is complex and possibly error prone. More importantly, any smart approach introduces failure modes. A particularly nasty problem arises when two clients are located behind the same NAT; this can happen when the NAT is installed by an ISP. In this case, even if the NAT is a cone NAT, we may have a problem, as exposed in the following diagram:

```
+-----------------------+
|                     |   Private network    |
| .-.                 |                     |
| src=9.0.0.1:4096    | -----               |
| (IPv4)              |                     |
| src=9.0.0.1:4097    | NAT                 |
| (Internet)<--------- | BOX                  |
| (UDPv4 tunneled)    | <--                 |
| '---'               | '-----'              |
| 10.0.0.2:1234       | 10.0.0.0.2:1234     |
| |                    | |                     |
| | 9.0.0.1            | |                     |
| | V                  | |                     |
| | Teredo             | |                     |
| | Server             | |                     |
| '-------------------' | '----------'
[IPv4-anycast]
```

The first client uses the private address and port 10.0.0.2:1234, which is mapped to 9.0.0.1:4096 by the NAT; the second client uses 10.0.0.3:1234, mapped to 9.0.0.1:4097. If the first client tries to send a direct packet to the second client, that packet will be routed to the address 9.0.0.1:4097, i.e. to the external IP address of the local NAT. What happens next is everybody’s guess: some NAT may do the right thing, some may drop the packet.

Our algorithm is designed to privilege robustness: unless the client knows otherwise, the packets will always be sent through a Teredo server, and will always be received through the single "pin-hole" that allows communication between the Teredo client and the Teredo anycast address. The packets will only be sent on the direct path if the client actually received a packet or a bubble on that direct path, demonstrating the existence of a chain of mappings on all the NAT located between the Teredo peers.

The algorithm for sending bubbles is designed to be conservative: we only send bubbles when necessary, and limit the transmission rate when we suspect that the sending of the bubbles is useless. In a type 3 NAT, we will send at most four bubbles to a destination, and then we will observe a silence period of 300 seconds.

6.3 Do we need the Refresh Interval Determination Procedure?

When the client is initialized, the Teredo Refresh Interval is set
to 30 seconds. This value is lower than the minimum interval found necessary in a measurement campaign conducted by a Microsoft team in January 2001; the measured values ranged between 45 seconds and more than 15 minutes. There is always a risk that some NAT manufacturers program some ever smaller time to live intervals for their mappings, but doing so would break many applications and would probably generate uproar from Internet users. By picking a conservatively small value, we guarantee that the service will work with most NATs.

On the other hand, picking a conservative value increases the maintenance traffic and the load on the Teredo servers. We know that in many cases interval as large as 5 or 10 minutes would be adequate; however, we also know that there is a high risk of false positives, e.g. when a NAT is connected by an ISDN "on demand" link. The determination procedure is designed to quickly find whether a value larger than 30 seconds is adequate, while not trying to achieve a value larger than 2 minutes. The parameters have been chosen for rapid convergence, i.e. at most 3 iterations between the initial value of 30 seconds and the maximum value of 120 seconds or 2 minutes.

6.4 Why do we use a Randomized Refresh Interval?

We specify in the maintenance procedure that the interval between successive refresh must be a random value chosen between 75% and 100% of the Teredo Refresh Interval. This randomization procedure is meant to avoid the possible risk of synchronization that is inherent to any periodic refresh mechanism; if synchronization occurred, all Teredo clients would send their router solicitation messages quasi simultaneously to the Teredo server, which would overwhelm the server. A synchronization phenomenon caused by periodic messages is studied in [SYNCHRO]; the 75%-100% interval is meant to meet the guidelines developed in this reference publication.

6.5 Scaling, failover and access control

The consequences of the use of anycast addresses on access control, scaling, and failover are discussed in [RFC3068] in the context of the "6to4" service.

Using an anycast address and stateless servers has beneficial consequences on scaling. It is possible to start deployment with a small number of Teredo servers, reachable across the Internet. It is then possible to deploy more servers across the Internet, as demand increases.

Teredo packets will be naturally forwarded to the nearest server, as specified in the IPv4 routing tables. If a Teredo server somehow breaks, or loses connectivity to the v6 Internet, it will cease to advertise reachability of the Teredo IPv4 anycast address. At that point, the local IGP will automatically compute a route towards the "next best" Teredo server. We expect that adequate monitoring tools
will be used to guarantee timely discovery of connectivity losses.

Only those ASes that run Teredo Servers and are willing to provide access to the v6 network announce a path to the Teredo IPv4 anycast prefix. They can use the existing structure of peering and transit agreements to control to whom they are willing to provide service, and possibly to charge for the service.

6.6 Why do we use a randomized link-local address?

The qualification and maintenance procedure could possibly be affected by a denial of service attack in which a malevolent third party sends a spoofed router advertisement to a Teredo Client, containing an IP address and UDP port different from the actual mapped IP address and mapped UDP port of the client. The attacker could learn the real values of the mapped address and port from the Teredo IPv6 Address used by the client; it could either predict the time at which the Teredo Client will issue a router solicitation or send relatively frequent advertisement messages.

The use of a randomized link-local address in the router solicitation messages provides some protection against this attack: a router advertisement whose destination address does not match the expected value will be silently discarded. The randomized address is used only once, which makes it hard to learn by a third party. Implementers should however make sure that the random values that they use are not easily guessable; it would be a good idea to use a secure random number generator, as discussed in [RFC1750].

6.7 What about firewalls?

The Teredo service is not designed to "transparently traverse firewalls." A local administrator can decide to allow or disallow the service, by programming the local firewall to authorize or deny traffic on the Teredo UDP port.

6.8 What about anycast routing and ingress filtering?

The Teredo specification allows multiple servers and relays to participate in the service with very little coordination. In practice, a set of coordinated servers and relays is defined by a Teredo service IPv6 prefix and a Teredo IPv4 address: the servers accept packets sent to the IPv4 anycast address and provide clients with IPv6 addresses that start with the IPv6 prefix; the relays advertise reachability in IPv6 of the same IPv6 prefix and send packets from the same IPv4 anycast address.

There are many reasons to allow for several servers and relays to share the same IPv4 anycast address and IPv6 prefix. The obvious reasons are reliability and load balancing: multiple stateless servers are more reliable than a single one; multiple relays scattered at many locations can handle the traffic load more.
efficiently. There are also some practical advantages to having one single large Teredo network: clients’ configuration is simpler and service reliability is maximized. Moreover, the direct Teredo routing communication between clients can only happen between clients of the same network: having one single large network maximizes the fraction of the traffic that can be routed directly.

However, relays and servers must use the anycast address as a source address, which triggers a dependency on address filtering: in order to be compatible with ingress filtering, the anycast addresses must be "topologically correct." Failure to do so would create the risk of a "black-hole" effect: a network administrator would turn on ingress filtering; packets sourced from the Teredo address would appear at unexpected interfaces; ingress filtering would cause these packets to be dropped.

In practice, the topological correctness is only achieved if the servers and relays are ready to receive packets from any destination to which they also send packets. Since the Teredo clients may send packet everywhere in the Internet, the set of destinations include the whole Internet. In order to avoid black-hole effects due to ingress filtering, the network administrator who establishes Teredo servers for the global Teredo network must be ready to advertise the reachability of the Teredo IPv4 anycast address to the whole IPv4 Internet. This will be in many cases a reasonable trade-off, since in practice the local servers will only receive traffic from the closest Teredo clients; the extra amount of traffic from non-local sources will be compensated by the increased efficiency of direct Teredo routing for the local clients, which will reduce the amount of traffic processed by the servers. However, we cannot impose this trade-off to all network operators all the time.

A network operator who is not ready to announce the service to the whole Internet has the option of operating a specific Teredo networks, using its own set of topologically correct addresses and prefixes. Some amount of reliability and efficiency can still be achieved if each specific network is served by an adequate number of servers and relays.

6.9 Why do we use the name Teredo?

"Teredo navalis" is the latin name little saltwater critter that is common in the harbors of warm seas and that digs worm holes in immersed wood pieces, such as boat hulls or pilings. The animal is not an actual worm - it is a mollusk. The Teredo service also digs holes, albeit in NATs, not in wood.

On one hand, one may think that the Teredo is a pretty nasty animal. On the other hand, the animal only survives in relatively clean and unpolluted water; its recent comeback in several Northern American harbors is a testimony to their newly retrieved cleanliness. The Teredo service should, in turn, contributes to a newly retrieved
Further work: tunnel service

It may be desirable in some cases to deploy stateful tunnel servers instead of the stateless Teredo servers. Tunnels servers generally require more resources, but an advantage is that they can potentially provide the users with "permanent" IPv6 addresses. It is possible to design a tunnel server protocol that is compatible with Teredo, in the sense that the same client could be used either in the Teredo service or with a tunnel service. The main requirements of such a solution would be:

1) Allow clients to be configured to send traffic to either the Teredo anycast address or the address of a specific tunnel server;

2) Modify the qualification procedure so that clients would detect whether they are connected to the Teredo service or to a tunnel server, e.g. by checking whether the advertised prefix is a Teredo prefix;

3) Ensure that clients, when they are connected in tunnel mode, do not send Teredo bubbles;

4) Ensure that clients, when they are connected in tunnel mode, send their packets to the tunnel server.

A tunnel server will manage a pool of /64 IPv6 prefixes, for which it will advertise IPv6 reachability. When it receives a router solicitation from a new client, the tunnel server will allocate one of the available prefixes to the client, and will keep track of the mapped IP address and mapped IP port of that client. When the tunnel server receives an IPv6 packet, it will check the table of prefixes and forward the packet over UDP to the corresponding mapped IP address and mapped UDP port of the client; the maintenance procedure performed by the client guarantees that these packets can be received through the NAT.

In order to fully specify the behavior of a tunnel server, we must study how the client could be authenticated and authorized, so that the same client would receive over time the same address, even if the mapping provided by the local NAT changes. It is probably possible to use the Authentication Header for this purpose, but the whole procedure needs to be properly analyzed.

Security Considerations

The security considerations for using a dynamic encapsulation of IPv6 over UDP are very similar to the implications of "6to4" [RFC3056], which we partially reproduce here. Implementers should be aware that, in addition to possible attacks...
against IPv6, security attacks against IPv4 must also be considered. Use of IP security at both IPv4 and IPv6 levels should nevertheless be avoided, for efficiency reasons. For example, if IPv6 is running encrypted, encryption of IPv4 would be redundant except if traffic analysis is felt to be a threat. If IPv6 is running authenticated, then authentication of IPv4 will add little. Conversely, IPv4 security will not protect IPv6 traffic once it leaves the Teredo service. Therefore, implementing IPv6 security is required even if IPv4 security is available.

In contrast with the 6to4 default, Teredo traffic will not be accepted and decapsulated from any source from which regular IPv4 traffic is accepted: the service is designed to prevent malevolent agent from using Teredo servers for "laundering" a denial of service attack. Teredo packets are always sent from either the Teredo IPv4 anycast address, or from an IPv4 address and UDP port that is strictly consistent with the encapsulated IPv6 source address; this MUST be checked by servers and clients, as specified in sections 5.2.2 and 5.3.1. We expect that only routers controlled by Internet Service Providers will be authorized to use the anycast address as a source address for UDP packets.

There is a possible denial of service attack against the address mapping service, which we discuss in section 5.8. In order to protect against this attack, we use a randomized link local address as source of router solicitations and check that the same random value is present in router advertisements.

In any case, any Teredo traffic whose source or destination address embeds a mapped IPv4 address which is not in the format of a global unicast address MUST be silently discarded by Teredo clients, servers, and routers as specified in sections 5.2.2, 5.2.3, 5.3.1, and 5.4. Specifically, this means that IPv4 addresses defined in [RFC1918], broadcast, subnet broadcast, multicast and loopback addresses are unacceptable.

There is a possible denial of service attack against the local peer discovery procedure, if attackers can manage to send spoofed local discovery bubbles to a Teredo client. The checks described in section 5.2.7 mitigate this attack. Clients who are more interested in security than in performance may decide to disable the local discovery procedure.

9  IANA Considerations

This memo documents a request to IANA to allocate a Teredo IPv6 service prefix, a Teredo IPv4 anycast prefix, a Teredo IPv4 anycast address and a Teredo UDP port.
1996], Section 10.4, and describes the applicable copyright for this document.

Copyright (C) The Internet Society July 12, 2001. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

11 Intellectual Property

The following notice is copied from RFC 2026 [Bradner, 1996], Section 10.4, and describes the position of the IETF concerning intellectual property claims made against this document.

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF’s procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary

Huitema

[Page 38]
rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

12 Acknowledgements

Many of the ideas in this memo are the result of discussions between the author and Microsoft colleagues, notably Brian Zill, John Miller and Rick Rashid. Several encapsulation details are inspired from early work by Keith Moore. The example in section 5.1 and a number of security precautions were suggested by Pekka Savola, and the discussion in 5.3 by Art Shelest. The local discovery procedure was suggested by Richard Draves and Dave Thaler. The document was reviewed by the NGTRANS working group; Brian Carpenter, Cyndi Jung, Keith Moore, Anssi Porttikivi, Pekka Savola, Eng Soo Guan and Mohit Talwar provided detailed comments.

13 References


Authors' Addresses

Christian Huitema
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Email: huitema@microsoft.com
Appendix A: Experimental client and server parameterization

Teredo clients and Teredo servers in the Global Teredo Network must be parameterized with the values of the Global Teredo IPv6 service prefix, Teredo IPv4 anycast address, and Teredo port. These parameters will eventually be assigned by the IANA. In the development and experimentation phase, these parameters are documented in the AAAA record associated with the DNS domain name: teredo.ipv6.microsoft.com.

There is a single AAAA record associated to the domain name. The record has the following format:

```
+----------+-------------+----+------------------------+---+
| Prefix   | IPv4 address| Port| 64 - 1 zero bits       | n |
+----------+-------------+----+------------------------+---+
```

The last 8 bits (120 to 127) in the address contain the length "1" of the experimental Teredo prefix; the 8 bits before last (112 to 119) encode the length of the experimental Global Teredo IPv4 anycast prefix.

The first "1" bits of the address encode the value of the experimental value of the Global Teredo prefix; the next 32 bits encode the experimental value of the Global Teredo IPv4 anycast address; the next 16 bits encode the experimental Teredo port.

The experimental values will be replaced by IANA assigned values as soon as such values are assigned.
Table of Contents:

1 Introduction .................................................... 1
2 Definitions ..................................................... 2
  2.1 Teredo service ............................................. 2
  2.2 Teredo Client .............................................. 2
  2.3 Teredo Server ............................................. 3
  2.4 Teredo Relay ............................................... 3
  2.5 Teredo IPv6 service prefix ................................ 3
    2.5.1 Global Teredo IPv6 service prefix ..................... 3
  2.6 Teredo IPv4 anycast prefix ................................ 3
    2.6.1 Global Teredo IPv4 anycast prefix ..................... 3
  2.7 Teredo IPv4 anycast address ................................ 3
    2.7.1 Global Teredo IPv4 anycast address .................... 3
  2.8 Teredo network ............................................. 3
  2.9 Teredo UDP port ........................................... 4
  2.10 Teredo bubble ............................................ 4
  2.11 Teredo service port ...................................... 4
  2.12 Teredo mapped address and Teredo mapped port .......... 4
  2.13 Teredo IPv6 client prefix ................................ 4
  2.14 Teredo IPv6 address ....................................... 4
  2.15 Teredo IPv6 anycast address ................................ 4
  2.16 Teredo Refresh Interval .................................. 4
  2.17 Teredo secondary port .................................... 4
  2.18 Random link-local address ................................ 5
  2.19 Teredo IPv4 Discovery Address ............................. 5
3 Design goals, requirements, and model of operation ............ 5
  3.1 Hypotheses about NAT behavior ............................ 5
    3.1.1 Types of UDP mappings ................................ 5
    3.1.2 Lifetime of UDP mappings .............................. 6
  3.2 IPv6 provider of last resort ................................ 7
    3.2.1 When to use Teredo? ................................... 7
    3.2.2 Autonomous deployment ................................ 7
    3.2.3 Minimal load on servers ............................... 7
    3.2.4 Automatic sunset ...................................... 7
  3.3 Operational Requirements .................................. 8
    3.3.1 Robustness requirement ................................ 8
    3.3.2 Protection against denial of service attacks .......... 8
    3.3.3 Protection against distributed denial of service attacks .... 8
    3.3.4 Non-requirement of permanent addresses ................. 8
    3.3.5 Compatibility with ingress filtering ................. 9
4 Model of operation and deployment ............................ 9
  4.1 Model of operation ......................................... 9
    4.1.1 Obtaining an address .................................. 10
    4.1.2 Sending a packet from a Teredo node to a regular IPv6 node .. 11
    4.1.3 Sending a packet from a regular IPv6 node to a Teredo node .. 13
    4.1.4 Exchanges between two Teredo nodes ..................... 14
  4.2 Deployment model .......................................... 15
    4.2.1 Deployment in the Global Teredo Network ............... 16
    4.2.2 Deployment of a specific Teredo network ............... 16
5 Specification of clients, servers and relays .................. 16
5.1 Message formats .............................................
5.1.1 Teredo IPv6 addresses ....................................
5.1.2 Teredo IPv6 packets encapsulation .....................
5.1.3 Maximum Transmission Unit .............................
5.1.4 Teredo bubble ...........................................
5.1.5 Server link local address ................................
5.1.6 Random link local address ...............................
5.1.7 Teredo local discovery bubbles .........................
5.2 Teredo Client specification ...............................