INTERNET DRAFT
<draft-ietf-ngtrans-shipworm-08.txt>
Expires March 17, 2003

C. Huitema
Microsoft

September 17, 2002

Teredo: Tunneling IPv6 over UDP through NATs

Status of this memo

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.

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 Translator (NAT) device: NATs 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 a 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 node 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 as follow. 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 5 contains the format of the messages and the specification of the protocol. Section 6 is a discussion of the key design choices. Section 7 presents the guideline for some further work that would be complementary to the current approach. Section 8 contains a security discussion, and section 9 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.

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 clients.

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:XXXX:/32.
(TBD IANA; experiments use the value 3FFE:831F::/32, taken from a range of experimental IPv6 prefixes assigned to Microsoft.)

2.6 Teredo UDP port

The UDP port number at which Teredo Servers are waiting for packets. The value of this port is 3544.

2.7 Teredo bubble

A Teredo bubble is a minimal IPv6 packet, made of an IPv6 header and a null payload - the payload type is set to 59, No Next Header, as per [RFC2460]. The Teredo clients and relays may send bubbles in order to create a mapping in a NAT.

2.8 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.

2.9 Teredo server address

The IPv4 address of the Teredo server selected by a particular user.

2.10 Teredo mapped address and Teredo mapped port

A global IPv4 address and a UDP port that results from the translation by one or several NATs 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.

2.11 Teredo IPv6 client prefix

A global scope IPv6 prefix composed of the Teredo IPv6 service prefix and the Teredo server address.

2.12 Teredo node identifier

A 64 bit identifier that contains the UDP port and IPv4 address at which a client can joined through the Teredo service, as well as a flag indicating the type of NAT through which the client accesses the IPv4 Internet.

2.13 Teredo IPv6 address

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

2.14 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 the combination of NATs in 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.

2.15 Teredo secondary port

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

2.16 Teredo IPv4 Discovery Address

An IPv4 multicast address used to discover other Teredo clients on 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 NATs, 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 UDP 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 devices 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 NATs 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 NATs implement either option 1 or option 2, i.e. cone NATs or restricted cone NATs. The Teredo solution ensures connectivity for clients located behind cone NATs, restricted cone NATs, or port-restricted cone NATs; it contains optimizations for clients located behind a cone NAT; it does not provide connectivity for clients located behind a symmetric NAT.

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 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 NATs, and the price of robustness is a reasonable amount of overhead, due to UDP encapsulation and transmission of bubbles. 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 configure the Teredo service as automatically

as possible. In practice, it is however required that the client learn the IPv4 address of a server that is willing to serve them;

some servers may also require some form of access control.

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. To achieve this objective, we request that servers only enable the packet exchange between clients, but do not carry the actual data packets: these packets will have to be exchanged directly between the Teredo clients, or through a destination-selected relay for exchanges between Teredo clients and other IPv6 clients.

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 systems 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 Minimal support cost

The service requires the support of servers and relays. In order to facilitate the deployment of these servers, the Teredo procedures are designed to minimize the fraction of traffic that has to be routed through the servers.

Meeting this objective implies that the Teredo addresses will incorporate the IPv4 address and UDP port through which a Teredo

client can be reached. This creates an implicit limit on the stability of the Teredo addresses, which can only remain valid as long as the underlying IPv4 address and UDP port remains valid.

3.3.3 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.

3.3.4 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.

3.3.5 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 Teredo specification must not force networks to disable 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 Teredo 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.

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

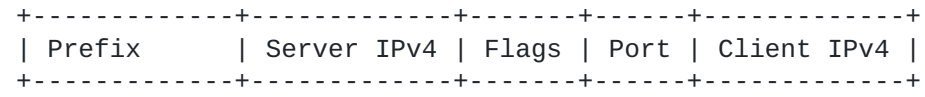
clients.

The primary role of the servers is to enable 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, bypassing the Teredo server.

4.1.1 Encoding of Teredo addresses

The Teredo addresses are composed of 5 components:

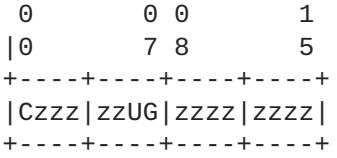


- Prefix: the 32 bit Teredo service prefix.
- Server IPv4: the IPv4 address of a Teredo server.
- Flags: a set of 16 bits that document type of address and NAT.
- Port: the obfuscated "mapped UDP port" of the Teredo service at the client
- Client IPv4: the obfuscated "mapped IPv4 address" of a client

In this format, both the "mapped UDP port" and "mapped IPv4 address" of the client are obfuscated. Each bit in the address and port number is reversed; this can be done by an exclusive OR of the 16-bit port number with the hexadecimal value 0xFFFF, and an exclusive OR of the 32-bit address with the hexadecimal value 0xFFFFFFFF.

A third party sends IPv6 packets to a Teredo client by sending these packets over UDP to the client IPv4 address and port if the server address is null, or if the third party has recently received direct traffic from the client. In the other cases, the third party will have to first synchronize with the client, by sending an initial bubble through the server.

The IPv6 addressing rules specify that "for all unicast addresses, except those that start with binary value 000, Interface IDs are required to be 64 bits long and to be constructed in Modified EUI-64 format." This dictates the encoding of the flags, 16 intermediate bits which should correspond to valid values of the most significant **16 bits of a Modified EUI-64 ID:**



In this format:

- The bits "UG" should be set to the value "00", indicating a non-global unicast identifier;
- The bit "C" (cone) should be set to 1 if the client believes it is behind a cone NAT, to 0 otherwise; these values determine

different server behavior during the qualification procedure, as specified in section 5.2.1, as well as different bubble processing by clients and relays.
- The bits indicated with "z" must be set to zero.

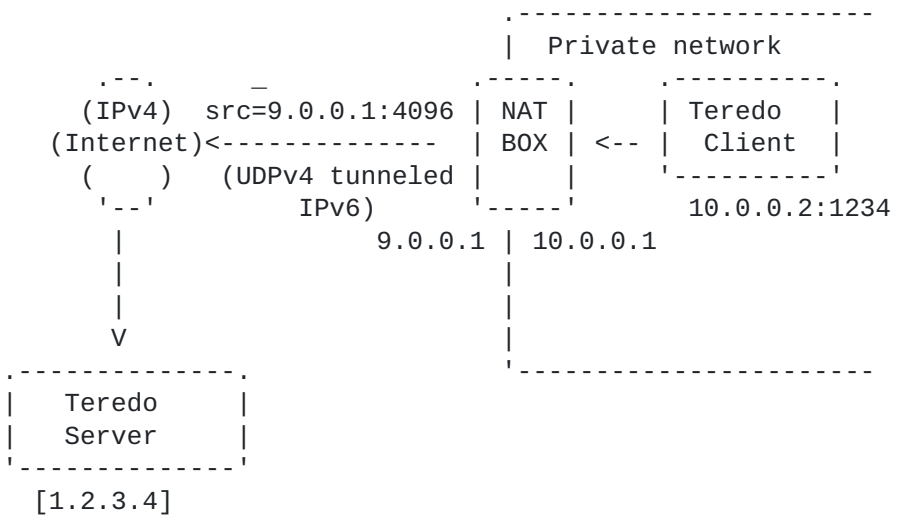
There are thus two valid values of the Flags field: "0x0000" (all null) if the cone bit is set to 0, and "0x8000" if the cone bit is set to 1.

In some cases, Teredo nodes use link-local addresses. These addresses contain a link local prefix (FE80::/64) and a 64 bit identifier, constructed using the same format as presented above. A difference between link-local addresses and global addresses is that the identifiers used in global addresses MUST include a global scope unicast IPv4 address, while the identifiers used in link-local addresses MAY include a private IPv4 address.

4.1.2 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 selects a Teredo server, and sends it a Router Solicitation message. The server replies with a router advertisement containing a Teredo prefix, composed of the Teredo service prefix and the IPv4 address of the server; the message also contains an "origin" indication that specifies the IPv4 address and port number from which the server received the router solicitation.

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. We present here a simplified version of the procedure, in which we are only concerned with determining the "mapping" of the client's address; in the next section, we will explain how this procedure is in fact combined with "cone NAT determination".



When the Teredo client is turned on, it sends an IPv6 router solicitation over UDP to the Teredo server, using the source address and ports 10.0.0.2:1234. The NAT intercepts 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 server receives the packet and notes the source address. It sends back a router advertisement that contains the server's prefix (e.g. PREF:0102:0304::/64); the advertisement also contains an origin indication that specifies the IPv4 address and UDP port from which the router advertisement was received, in this case 9.0.0.1:4096. The router sends the router advertisement back over UDP to the mapped IPv4 address 9.0.0.1:4096. The packet will have the following format:

IPv4	UDP	Origin indication	IPv6 RA

The "origin indication" field specifies the "mapped IPv4 address" and "mapped port" of the client; the IPv6 router advertisement specifies the prefix announced by the server.

The IPv4 packet is received by the NAT, which has remembered the mapping between this external address and port pair and the private address and port pair, 10.0.0.2:1234; the NAT forwards the packet to the Teredo client.

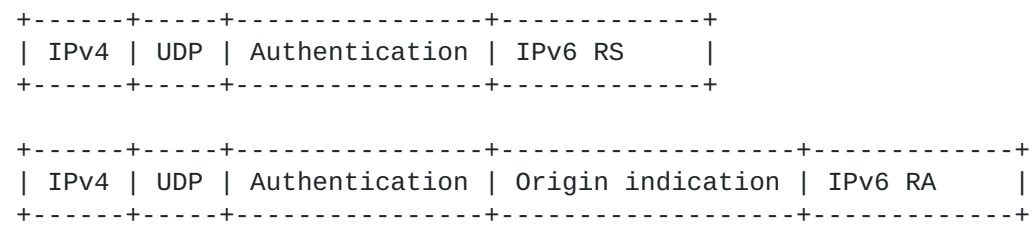
Upon reception of this prefix, the client composes a Teredo address, which in our example will be:

PREF:102:304::EFFF:F6FF:FFFE

In this address, "PREF" is the Teredo IPv6 service prefix of length

32; "102:304" is the hexadecimal notation of the address of the Teredo server, in this example 1.2.3.4 (expressed with leading zeros as 0102:0304); "EFFF" is equal to the XOR of "FFFF" with "1000", which is the hexadecimal notation of the mapped port "4096"; and "F6FF:FFFE" is equal to the XOR of "FFFF:FFFF" with "0900:0001", which is the hexadecimal notation of the IPv4 mapped address "9.0.0.1".

In some environments, it is necessary to secure this exchange, to avoid the risk of an attacker "spoofing" the server's return. In these environments, the client will be provisioned with a secret key and a key identifier; the key is shared with the server. Both the client's solicitation and the server's response will be protected by an authentication token, carried in the UDP message:



The authentication token uses the secret key and the key identifier to guarantee the integrity and the authenticity of the packets.

4.1.3 Determining the type of NAT

The previous section presented a simplified version of the prefix assessment procedure. For better efficiency, the client needs to determine the type of NAT behind which it is located: clients located behind a "cone" NAT can receive traffic directly, and we want to take advantage of this optimization: if the client is located behind a cone NAT, it should use Teredo addresses in which the "cone" bit is set to 1.

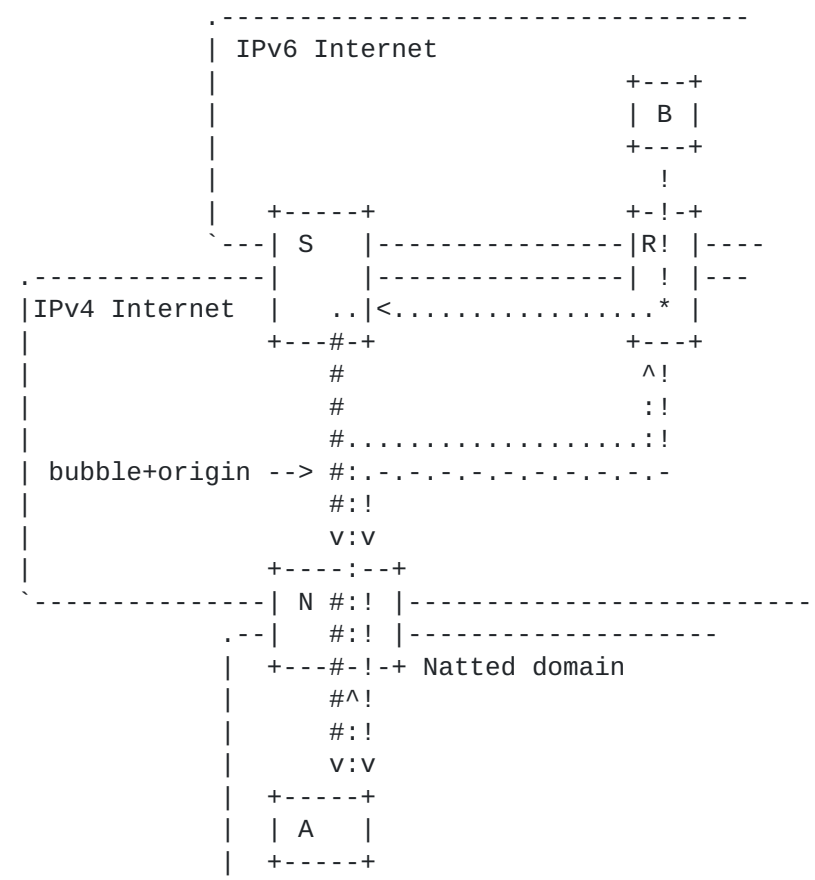
To achieve this result, the client first sends a router solicitation message from a link-local address in which the "cone" bit is set to 1; when the server observes that the cone bit is set, it sends the router advertisement in a UDP packet from a "secondary" IPv4 address, i.e., a different IPv4 address than the "server address" to which the solicitation was sent. Only nodes located behind a cone NAT will be able to receive this reply; in consequence, if the client receives this reply, from an IPv4 address different than the server address, it determines that it is located behind a cone NAT.

If the client does not receive a response to the first solicitation, it repeats the procedure and sends a solicitation message from a link-local address in which the "cone" bit is set to 0; the server will send the reply from its "nominal" IPv4 address, so the answer can be received by a non-cone NAT. The client will assume that it is not behind a cone NAT.

The Teredo service does not work if the client is located behind a symmetric NAT. The client must thus do an extra step, after receiving the router advertisement from the server nominal address: it should repeat the procedure by sending a solicitation to a secondary server IPv4 address, and compare the mapped addresses and mapped port in the two replies. If they are not the same, the client detects that it is behind a symmetric NAT and cannot use the Teredo service.

4.1.4 First packet from an IPv6 node to a Teredo node

The transmission of packets between a regular IPv6 node and a Teredo node is presented in the following diagram, in which "A" is a Teredo client located behind the NAT "N", "S" the Teredo server chosen by "A", "B" a regular IPv6 node, and "R" a Teredo Relay close to "B" in the IPv6 Internet Topology.



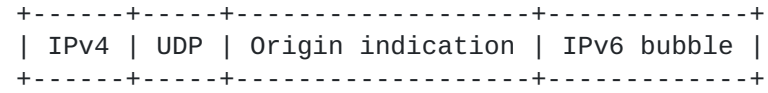
We assume that A has already established its Teredo address through an RA/RS exchange with S, as explained in section 4.1.2; in this example, we will assume that the cone NAT determination failed. We will assume that the results of these exchanges are the following:

- A's private 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:102:304::EFFF:F6FF:FFFE
- The server's IPv4 address and port are: 1.2.3.4:3544
- The relay's IPv4 address and port are: 5.6.7.8:3544
- B's IPv6 address is: 2000::B

When B sends a packet to A, B simply follows IPv6 rules. The packet is forwarded to the nearest relay, R, over IPv6. R examines the destination address, and observes that the address includes the notation of a server, 1.2.3.4. Since R has not yet received any direct traffic from A, and since A is not located behind a cone NAT, R cannot send the packet directly to A: it would probably be rejected by the NAT N. Instead, R queues the packet, and formats a bubble that it forwards towards A over UDP, to the address of S: 1.2.3.4:3544.

When S receives the bubble, it notices that the source is not a Teredo client, but instead a regular IPv6 address. S forwards the packet to A using a special envelope. The packet will have the following format:



In the IPv4 and UDP header, the source will be set to the server's address and port, 1.2.3.4:3544, and the destination to A's mapped address, as indicated in the IPv6 destination address: 9.0.0.1:4096. The origin indication element will carry the IPv4 address and UDP port of R: 5.6.7.8:3544.

The NAT N will receive this message. It will use the existing mapping to rewrite 9.0.0.1:4096 to 10.0.0.2:1234, and forward the packet to A. When A receives the packet, it will immediately send a bubble back to the origin of the bubble, i.e. towards R, at the address 5.6.7.8:3544.

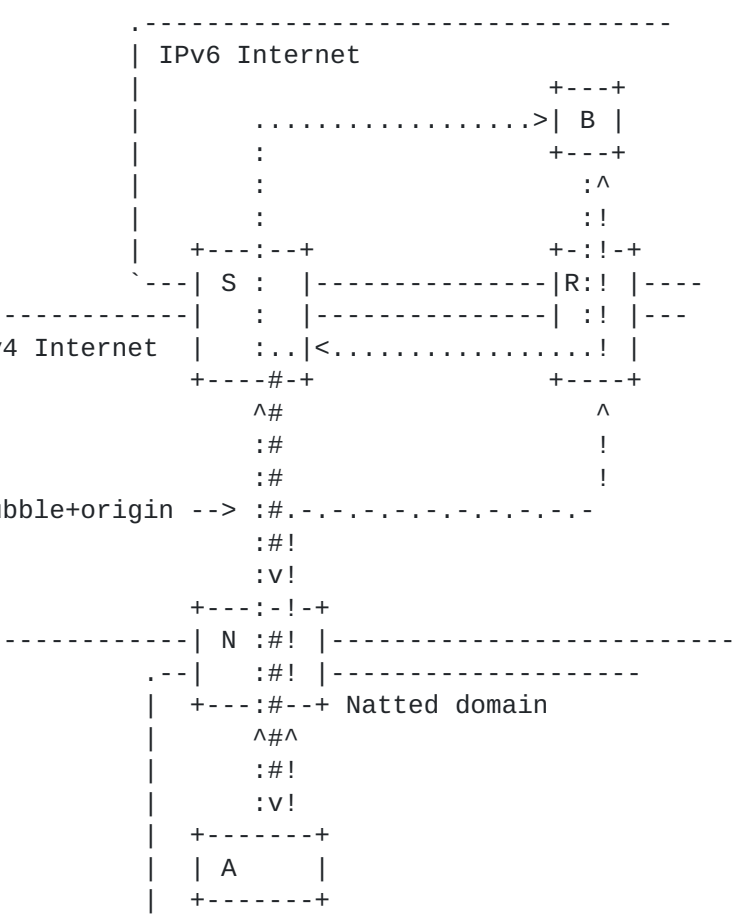
When R receives the bubble from A, it notes that direct transmission towards A is now possible. It forwards the queued packet to the mapped address of A, 9.0.0.1:4096. The packet is received by the NAT; since A has recently sent a packet to R, a mapping exists and the packet is forwarded to A.

If the cone NAT validation had been successful, A would have used the IPv6 address PREF:102:304:8000:EFFF:F6FF:FFFE, and R would have sent B's packet directly to A.

4.1.5 First 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 chosen by

"A", "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 section 4.1.2; in this example, we will assume that the client is not located behind a cone NAT. We will assume that the results of these exchanges are the following:

- A's private 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:102:304::EFFF:F6FF:FFFE
- The server's IPv4 address and port are: 1.2.3.4:3544
- The relay's IPv4 address and port are: 5.6.7.8:3544
- B's IPv6 address is: 2000::B

A has to transmit an IPv6 packet to B. A's first action is to learn the address of the relay R closest to B. To do so, A sends an ICMPv6 "echo request" toward B. This request carries the source address PREF:102:304::EFFF:F6FF:FFFE (A's address), and the destination address 2000::B (B's address); the Data field of the echo request carries a nonce value, chosen by A. The request is encapsulated by A in a UDP datagram, from source address and port 10.0.0.2:1234, to

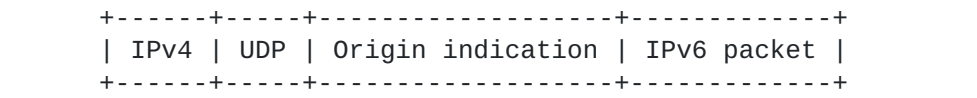
destination address and port 1.2.3.4:3544.

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 the 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:102:304::EFFF:F6FF:FFFE.

When the request is received, B sends the echo reply to A; B simply follows IPv6 routing rules. The packet is forwarded to the nearest relay, R, over IPv6. R examines the destination address, and observes that the address includes the address of a server, 1.2.3.4. Since R has not received any direct traffic from A, R forwards the packet over UDP to the address of S: 1.2.3.4:3544.

When S receives the packet, it notices that the source is not a Teredo client, but instead a regular IPv6 address. S forwards the packet to A using a special envelope. The packet will have the following format:



In the IPv4 and UDP header, the source will be set to the server's address and port, 1.2.3.4:p, and the destination to A's mapped address, as indicated in the IPv6 destination address: 9.0.0.1:4096. The origin indication element will carry the IPv4 address and UDP port of R: 5.6.7.8:3544.

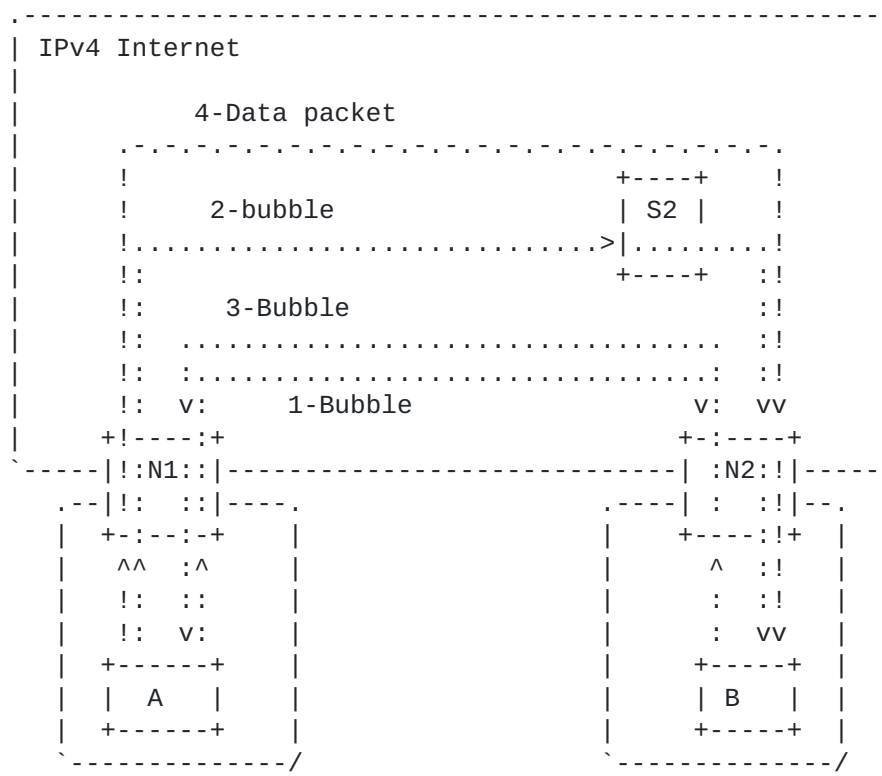
The NAT N will receive this message. It will use the existing mapping to rewrite 9.0.0.1:4096 to 10.0.0.2:1234, and forward the packet to A. When A receives the packet, it will notice the origin indication, as reported by the server, and it will have learned that B can possibly be reached through the relay address 5.6.7.8:p. A can also note that the Data field of the ICMPv6 echo reply matches the nonce that was previously sent, which provides a reasonable assurance that the packet does in fact come from B. At that point, A can send the original data packet to B, by encapsulating it in a UDP datagram bound to the IPv4 address and UDP port of R: 5.6.7.8:3544; R will then normally relay the packet to B. Once the knowledge of R's address has been acquired, A will send all further packets directly through R, without having to repeat the ICMP exchange.

If the cone NAT validation had been successful, A would have used the IPv6 address PREF:102:304:8000:EFFF:F6FF:FFFE, and R would have sent B's reply directly to A. In that case, A would have learned R's address from the IPv4 source address and UDP source port of the incoming packet.

Finally, we may observe that this procedure supposes that the relay will send the ICMP echo reply to a non-cone client directly through the server. This is a slight variation of the procedure described in the previous section, in which the IPv6 packets are queued waiting for completion of a bubble exchange between client and relay. The relay may in fact choose to queue the ICMPv6 echo reply, just like it queues normal packets, and forward the ICMPv6 echo reply when the bubble exchange is complete. In that case, the client will learn the relay's address from the IPv4 source address and UDP source port of the incoming packet.

4.1.6 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 S2, chosen by B. We will assume that the NAT N2 is a "restricted cone" or "port-restricted cone" NAT; in this example, we will also assume that A is behind a restricted cone or port-restricted cone NAT.



We will assume that A and B have already obtained Teredo addresses by RS/RA exchanges with they respective servers S1 and S2, and they have made sure that they can receive packets from these respective servers, for example by sending a keepalive packet every minute. In the example, we will assume the following values:

- A's private address and port are: 10.0.0.2:1234.
- A's mapped address and port are: 9.0.0.1:4096.
- A is served by S1, whose address is: 1.2.3.4
- A's IPv6 address has been set to PREF:102:304::EFFF:F6FF:FFFE
- B's private address and port are: 10.0.0.3:3456.
- B's mapped address and port are: 8.0.0.1:1024.
- B is served by S2, whose address is: 9.10.11.12
- B's IPv6 address has been set to PREF:90A:B0C::FBFF:F7FF:FFFE

A has learned B's address, for example from the DNS, and starts UDP or TCP communication with B. The first data packet will be sent from A's IPv6 address (PREF:102:304::EFFF:F6FF:FFFE) to B's IPv6 address (PREF:90A:B0C::FBFF:F7FF:FFFE). Since A has no prior knowledge of B, and since B is not located behind a cone NAT, A cannot send the packet directly to B; the packet is queued. Instead, A prepares two bubbles, from IPv6 source address PREF:102:304::EFFF:F6FF:FFFE to IPv6 destination address PREF:90A:B0C::FBFF:F7FF:FFFE.

The first bubble is sent from A to the mapped IPv4 address of B, 8.0.0.1:1024. It will probably not be received by B, since there is no establish mapping at the NAT N2. The purpose of this bubble is only to establish a mapping in the NAT N1.

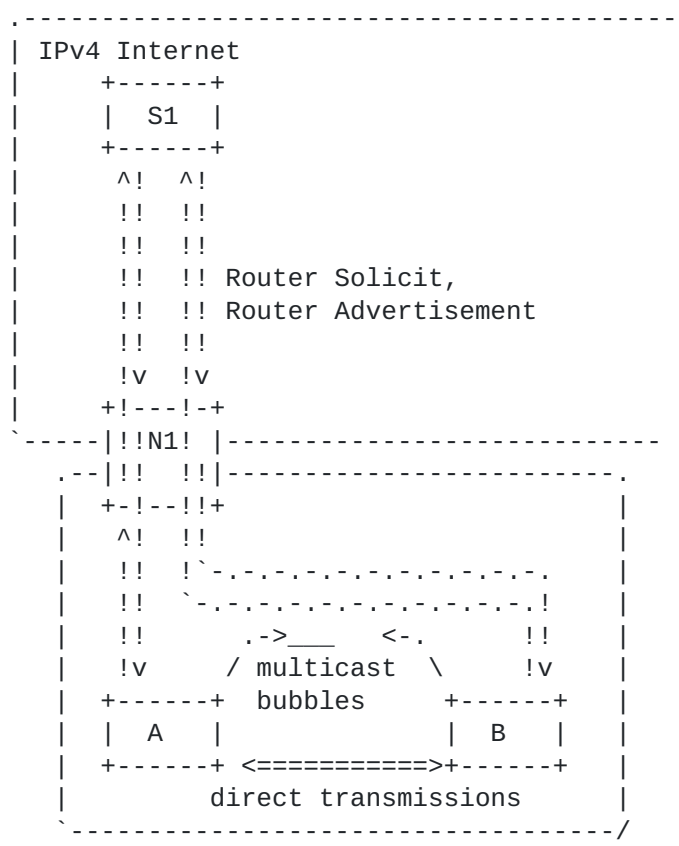
The second bubble is sent from A to S2's address, 9.10.11.12:3544. The bubble passes through N1. S2 receives it and transmits it to B. Since there is an established mapping in N2 for transmission between B and S2, the bubble is forwarded to B.

B responds to this bubble with its own bubble, from IPv6 source address PREF:90A:B0C::FBFF:F7FF:FFFE to IPv6 destination address PREF:102:304::EFFF:F6FF:FFFE. This response bubble is sent directly to the mapped address of A, 9.0.0.1:4096. Since a mapping was established by the first bubble, this third bubble is received by A.

When A receives the third bubble, it knows that direct communication with B is now possible. The queued packet can now be directly transmitted.

4.1.7 Exchanges between two Teredo nodes on the same link

The following diagram shows two Teredo clients, A and B, connected to the same link, which is connected to the Internet through the NAT N1. The exchanges will use the Teredo server S1. We are not making any assumption about the NAT N1. This scenario explains how the exchanges between clients on the same link can be optimized to avoid routing all packets through the server S1.



Both A and B discover their Teredo prefix by interacting with the server S1, as explained in 4.1.2.

In order to enable direct transmission on the local link, both A and B send Teredo bubbles to the Teredo IPv4 Discovery Address, an IPv4 multicast address whose scope is limited to the local link. These bubbles enable the local hosts to discover the local IPv4 address and the local UDP port associated with the Teredo IPv6 address of a local host. The data packets can then be sent over UDP to this address, avoiding the long path through the server S1.

4.2 Deployment model

The deployment model makes three assumptions:

- That all clients, servers and relays will be cognizant of the Teredo service prefix and the Teredo port,

- That the clients will be configured with the IPv4 address of a server,

- That there will be an adequate deployment of Teredo relays.

4.2.1 Server deployment

Servers may be deployed either as part of an ISP offer to its subscribers, or as an enabler for an application that requires direct IPv6 communication between client hosts. Servers are expected to perform some amount of access control; for example, an ISP server may refuse to serve requests that don't originate from an address within this ISP's network; a server set up by an application provider may require that clients provide some form of proof that they are actually using the application.

Servers will originate IPv6 packets whose source address will be the Teredo IPv6 address of one of their clients. In order to abide with IPv6 ingress filtering rules, servers should only do so if, as an IPv6 router, they advertise reachability of the Teredo service prefix.

4.2.2 Relay deployment

The ingress filtering rule implies that all Teredo servers should be able to act as Teredo relays; however, there is no requirement that all Teredo relays act as Teredo servers. The only deployment requirement for Teredo relays is IPv4 connectivity, and the capacity to advertise a route to the Teredo service. There can be three kinds of Teredo relays:

- Globally accessible Teredo relays announce reachability of the Teredo service to the whole Internet, e.g. by means of BGP.

- Domain-specific Teredo relays announce reachability of the Teredo service to a specific domain, e.g. by means of IGP.

- Host-specific Teredo relays announce reachability of the Teredo service within a single host.

In the initial deployment of the Teredo service, we expect to find a small number of globally accessible relays. We also expect that, if the service is deployed to enable a specific application, all the hosts that participate in the application and have adequate IPv4 access will implement a host-based Teredo relay.

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.

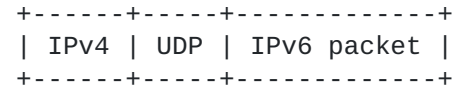
Teredo clients, servers and relays must implement the sunset procedure defined in section 5.5.

5.1 Message formats

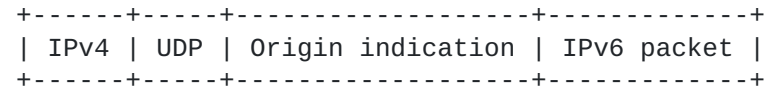
5.1.1 Teredo IPv6 packets encapsulation

Teredo IPv6 packets are transmitted as UDP packets [RFC768] within IPv4 [RFC791]. The source and destination IP addresses and UDP ports take values that are specified in this section. Packets can come in one of two formats, simple encapsulation and encapsulation with origin indication.

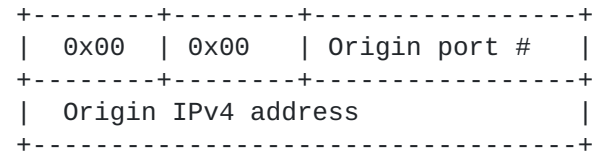
When simple encapsulation is used, the packet will have a simple format, in which the IPv6 packet is carried as the payload of a UDP datagram:



When relaying packets received from third parties, the server may insert an origin indication in the first bytes of the UDP payload:



The origin indication encapsulation is an 8-octet element, with the following content:



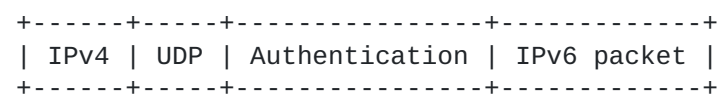
The first two octets of the origin indication are set to a null value; this is used to discriminate between the simple encapsulation, in which the first 4 bits of the packet contain the indication of the IPv6 protocol, and the origin indication.

The following 16 bits contain the obfuscated value of the port number from which the packet was received, in network byte order. The next 32 bits contain the obfuscated IPv4 address from which the packet was received, in network byte order. In this format, both the original "IPv4 address" and "UDP port" of the client are obfuscated. Each bit in the address and port number is reversed; this can be

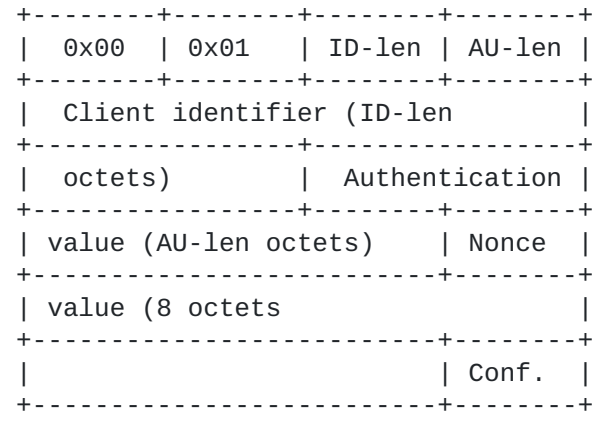
done by an exclusive OR of the 16-bit port number with the hexadecimal value 0xFFFF, and an exclusive OR of the 32-bit address with the hexadecimal value 0xFFFFFFFF.

For example, if the original UDP port number was 337 (hexadecimal 0151) and original IPv4 address was 1.2.3.4 (hexadecimal: 01020304), the origin indication would contain the value "0000FEAEFEFDFCFB".

When exchanging Router Solicitation and Router Advertisement messages between a client and its server, the packets may include an authentication parameter:

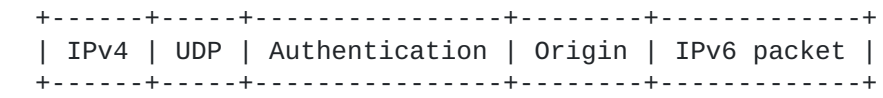


The authentication encapsulation is a variable length-element, containing a client identifier, an authentication value, a nonce value, and a confirmation byte.



The first octet of the authentication encapsulation is set to a null value, and the second octet is set to the value 1; this enables differentiation from IPv6 packets and from origin information indication encapsulation. The third octet indicates the length of the client identifier; the fourth octet indicates the length of the authentication value. The computation of the authentication value is specified in section 5.2.2. The authentication value is followed by an 8-octet nonce, and by a confirmation byte.

Authentication and origin indication encapsulations may sometimes be combined, for example in the RA responses sent by the server. In this case, the authentication encapsulation MUST be the first element in the UDP payload:



5.1.2 Maximum Transmission Unit

Since Teredo uses UDP as an underlying transport, a Teredo Maximum Transmission Unit (MTU) could potentially be as large as the payload of the largest valid UDP datagram (65507 bytes). However, since Teredo packets can travel on unpredictable paths over the Internet, it is best to contain this MTU to a small size, in order to minimize the effect of IPv4 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 Don't Fragment (DF) bit of the encapsulating IPv4 header.

5.2 Teredo Client specification

Before using the Teredo service, the client must be configured with:

- the IPv4 address of a server.

If secure discovery is required, the client must also be configured with:

- a client identifier,
- a secret value, shared with the server.

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,
- Teredo IPv6 prefix associated with the Teredo service port,
- Teredo IPv6 address or addresses derived from the prefix,
- Random link local address,
- Date and time of the last interaction with the Teredo server,
- Teredo Refresh Interval,
- Randomized Refresh Interval,
- List of recent Teredo peers.

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; it should then perform the cone NAT determination procedure,

which determines the cone NAT status and may alter the value of the 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 IPv6 address of the peer,
- The mapped IPv4 address and mapped UDP port of the peer,
- The status of the mapped address, i.e. trusted or not,
- The value of the last "nonce" sent to 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.

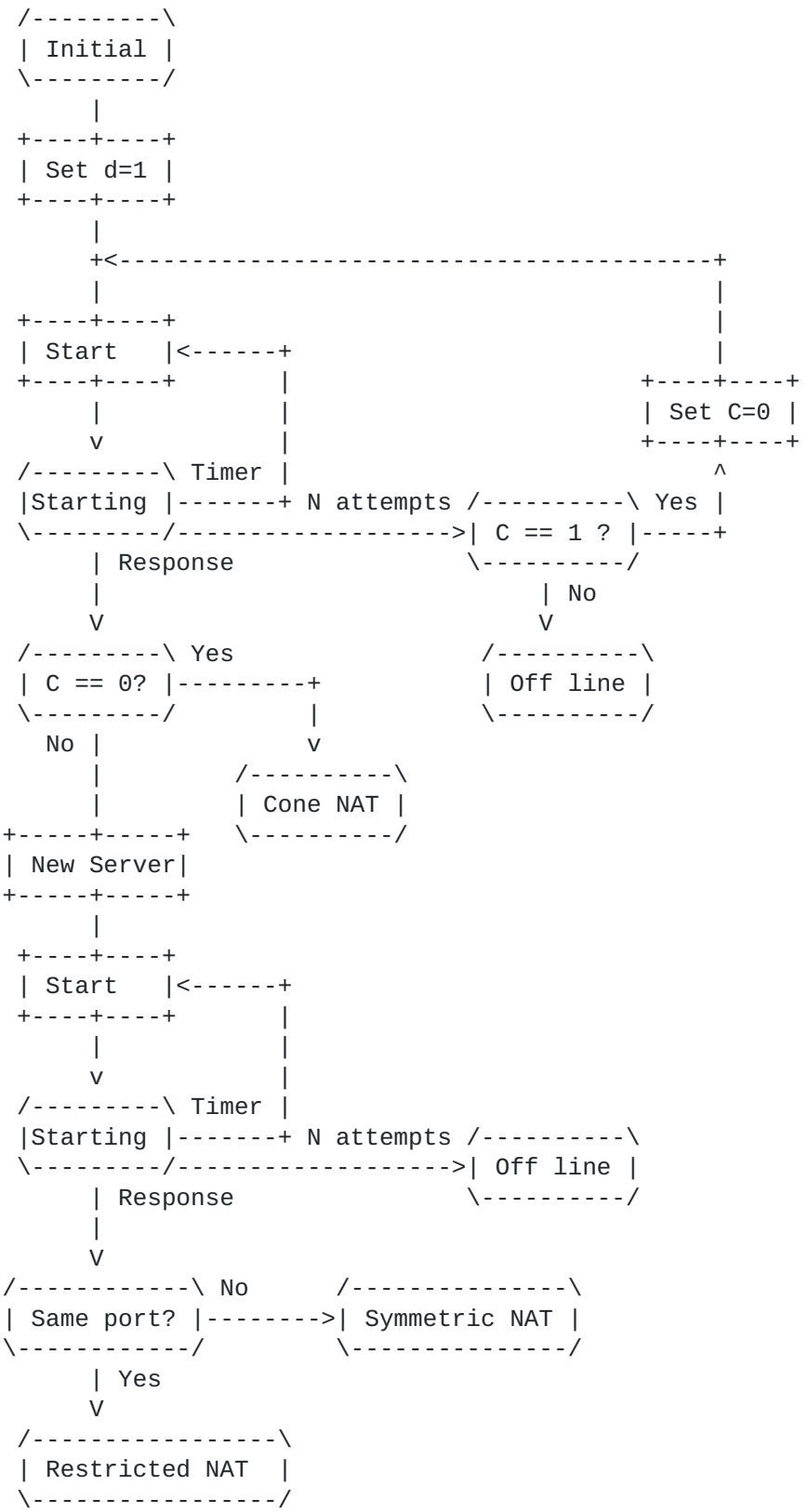
The list of peers is used to enable the transmission of IPv6 packets by using a "direct path" for the IPv6 packets. The list of peers could grow over time. Clients should implement a list management strategy, for example deleting the least recently used entries. Clients should make sure that the list has a sufficient size, to avoid unnecessary exchanges of bubbles.

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 5.2.8.

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, and in an "off-line" state if unsuccessful.


Initial
Set d=1
Start
Timer
Starting
N attempts
Yes
C == 1 ?
Response
No
Set C=0
C == 0?
Yes
No
Off line
Cone NAT
New Server
Start
Timer
Starting
N attempts
Off line
Response
Same port?
No
Symmetric NAT
Yes
Restricted NAT

Initially, the Teredo connectivity status is set to "Initial".

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 link-local address and uses it as the IPv6 source of the message; the "cone" bit in the address is set to 1; 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 server's IPv4 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 Solicitation message. If no response has arrived after N repetitions, the client concludes that it is not behind a cone NAT. It sets the "cone" bit to 0, and repeats the procedure. If after N other timer expirations and retransmissions there is still no response, 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=4 seconds, and the maximum number of repetitions is set to N=3.

If a response arrives, the client checks that the response contains an origin indication and a valid router advertisement as defined in [RFC2461], that the IPv6 destination address is equal to the link-local address used in the router solicitation, and that the router advertisement contains exactly one advertised Prefix Information option. This prefix should be a valid Teredo IPv6 server prefix: the first 32 bits should contain the global Teredo IPv6 service prefix, and the next 32 bits should contain the server's IPv4 address. If this is the case, the client learns the Teredo mapped address and Teredo mapped port from the origin indication. The source address of the Router Advertisement is a link-local server address of the Teredo server. (Responses that are not valid advertisements are simply discarded.)

If the client has received an RA with the "Cone" bit set to 1, it is behind a cone NAT and is fully qualified. If the RA is received with the Cone bit set to 0, the client does not know whether the local NAT is restricted or symmetric. The client selects a secondary IPv4 server address, and repeats the procedure, the cone bit remaining to the value zero. If the client does not receive a response, it detects that the service is not usable. If the client receives a response, it compares the mapped address and mapped port in this second response to the first received values. If the values are different, the client detects a symmetric NAT: it cannot use the Teredo service. If the values are the same, the client is detects a restricted cone NAT: the client is qualified to use the service.

If the client is qualified, it builds a Teredo IPv6 address using

the Teredo IPv6 server prefix learned from the RA and the obfuscated values of the UDP port and IPv4 address learned from the origin indication. The cone bit should be set to the value used to receive the RA, i.e. 1 if the client is behind a cone NAT, 0 otherwise. The client can start using the Teredo service.

5.2.2 Secure qualification

The client may be required to perform secured qualification. The client will perform exactly the algorithm described in 5.2.1, but it will incorporate an authentication encapsulation in the UDP packet carrying the router solicitation message, and it will verify the presence of a valid authentication parameter in the UDP message that carries the router advertisement provided by the sender.

In these packets, the nonce value is chosen by the client, and is repeated in the response from the server; the client identifier is a value with which the client was configured. The confirmation byte is set to 0 by the client. A null value returned by the server indicates that the client's key is still valid; a non-null value indicates that the client should obtain a new key.

The authentication value is computed according to the HMAC specification [RFC2104] using the following specifications:

- the hash function shall be the MD5 function [RFC1321].
- the secret value shall be the shared secret with which the client was configured

The clear text to be protected includes:

- the nonce value,
- the confirmation byte,
- the origin indication encapsulation, if it is present,
- the IPv6 packet.

If the HMAC verification fails, the packet is silently discarded.

5.2.3 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."

When a UDP packet is received over the Teredo service port, the Teredo client checks that it is encoded according to the packet encoding rules defined in 5.1.1, and that it contains either a valid IPv6 packet as specified in [RFC2460], or the combination of a valid origin indication encapsulation and a valid IPv6 packet, possibly protected by a valid authentication encapsulation. If this is not the case, the packet is silently discarded.

Then, the Teredo client examines the IPv4 source address and UDP port number from which the packet is received. If these values match the IPv4 address of the server and the Teredo port, the client updates the "date and time of the last interaction with the Teredo server" to the current date and time; if an origin indication is present, the client should perform the "direct IPv6 connectivity test" described in section 5.2.9.

If the values are different, the client examines the IPv6 source address of the packet:

1) If there is an entry for the source IPv6 address in the list of peers whose status is trusted, the client compares the mapped IPv4 address and mapped port in the entry with the source IPv4 address and source port of the packet. If the values match, the packet should be accepted; the date and time of the last reception from the peer should be updated.

2) If there is an entry for the source IPv6 address in the list of peers whose status is not trusted, the client checks whether the packet is an ICMPv6 echo reply. If this is the case, and if the content of the reply matches the "nonce" stored in the peer entry, the packet should be accepted; the status of the entry should be changed to "trusted", the mapped IPv4 and mapped port in the entry should be set to the source IPv4 address and source port from which the packet was received, and the date and time of the last reception from the peer should be updated; any packet queued for this IPv6 peer should be de-queued and forwarded to the newly learned IPv4 address and UDP port.

3) If the source IPv6 address is a Teredo address, the client compares the mapped IPv4 address and mapped port in the source address with the source IPv4 address and source port of the packet. If the values match, the client MUST create a peer entry for the IPv6 source address in the list of peers; it should update the entry if one already existed; the mapped IPv4 address and mapped port in the entry should be set to the value from which the packet was received, and the status should be set to "trusted". If a new entry is created, the last transmission date is set to 30 seconds before the current date, and the number of bubbles to zero. If the packet is a bubble, it should be discarded after this processing; otherwise, the packet should be accepted. In all cases, the client must de-queue and forward any packet queued for that destination.

4) If the IPv4 destination address through which the packet was received is the Teredo IPv4 Discovery Address, the source address is a valid Teredo address, and the destination address is the "all nodes on link" multicast address, the packet should be treated as a local discovery bubble. The client SHOULD create a new peer entry for the IPv6 source address; the mapped IPv4 address and mapped port in the entry should be set to the value from which the packet was

received, and the status should be set to "trusted". However, clients MAY decide to only accept local discovery bubbles if the mapped IPv4 address included in the IPv6 source address is the same as the mapped IPv4 address of the client.

5) In the other cases, the packet may be accepted, but the client should be conscious that the source address may be spoofed; before processing the packet, the client should perform the "direct IPv6 connectivity test" described in section 5.2.9.

Whatever the IPv4 source address and UDP source port, the client that receives an IPv6 packet MAY send a Teredo bubble towards that target, as specified in section 5.2.6.

5.2.4 Packet transmission

When a Teredo client has to transmit a packet over a Teredo interface, it examines the destination IPv6 address. The client checks first if there is an entry for this IPv6 address in the list of recent Teredo peers, and if the entry is still valid: an entry associated with a local peer is valid if the 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 5.2.8.)

The client then performs the following:

1) If there is an entry for that IPv6 address in the list of peers, and if the status of the entry is set to "trusted", the IPv6 packet should be sent over UDP to the mapped IPv4 address and mapped UDP port of the entry. The client updates the date of last transmission in the peer entry.

2) If the destination is not a Teredo IPv6 address, the packet is queued, and the client performs the "direct IPv6 connectivity test" described in section 5.2.8. The packet will be de-queued and forwarded if this procedure completes successfully. If the direct IPv6 connectivity test fails to complete within a 2 second time-out, it should be repeated up to 3 times.

3) If the destination is a Teredo IPv6 address in which the cone bit is set to 1, the packet is sent over UDP to the mapped IPv4 address and mapped UDP port extracted from that IPv6 address.

4) If the destination is a Teredo IPv6 address in which the cone bit is set to 0, the packet is queued. If the client is not located behind a cone NAT, it sends a direct bubble to the Teredo destination, i.e. to the mapped IP address and mapped port of the destination. In all cases, the client sends an indirect bubble to

the Teredo destination, sending it over UDP to the server address and to the Teredo port. The packet will be de-queued and forwarded when the client receives a bubble or another packet directly from this Teredo peer. If no bubble is received within a 2 second time-out, the bubble transmission should be repeated up to 3 times.

In cases 3 and 4, before sending a packet over UDP, the 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. (Note that a packet can legitimately be sent to a non-global unicast address in case 1, as a result of the local discovery procedure.)

5.2.5 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 server, as specified in 5.2.1; the client should use the same value of the "cone" bit that resulted in the reception of an RA during the qualification procedure.

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 mapped address and port correspond to the current Teredo address. If this is not the case, the mapping has changed; the client must mark the old address as invalid, and start using the new address.

5.2.6 Sending Teredo Bubbles

The Teredo client may have to send a bubble towards another Teredo client, either after a packet reception or after a transmission attempt, as explained in sections 5.2.3 and 5.2.4.

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 IPv6 address in the current list of peers. If there is no entry, the client MUST create a new list entry for the address, 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 is 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 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 2 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. 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.

5.2.7 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.

5.2.8 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 through IPv4 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 local discovery bubbles to the Teredo IPv4 Discovery Address at random intervals, uniformly distributed between 200 and 300 seconds. A local Teredo bubble has the following characteristics:

- 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

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 the filtering rules described in section 5.2.5, and also by "link only" multicast scope of the Teredo IPv4 Discovery 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 do 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.

5.2.9 Direct IPv6 connectivity test

The Teredo procedures are designed to enable direct connections between a Teredo host and a Teredo relay. Teredo hosts located behind a cone NAT will receive packets directly from relays; other

Teredo hosts will learn the original addresses and UDP ports of third parties through the local Teredo server. In all of these cases, there is a risk that the IPv6 address of the source be spoofed by a malevolent party. Teredo hosts must make two decisions, whether to accept the packet for local processing, and whether to transmit further packets to the IPv6 address through the newly learned IPv4 address and UDP port. The basic rule is that the hosts should be generous in what they accept, and careful in what they send. Refusing to accept packets due to spoofing concerns would compromise connectivity, and should only be done when there is a near certainty that the source address is spoofed; on the other hand, sending packets to the wrong address should be avoided.

When it wants to send a packet to an IPv6 node on the IPv6 Internet, the client should check whether a valid peer entry already exists for the IPv6 address of the destination. If this is not the case, the client will pick a random number (a nonce) and format an ICMPv6 Echo Request message whose source is the local Teredo address, whose destination is the address of the IPv6 node, and whose Data field is set to the nonce. The nonce value and the date at which the packet was sent will be documented in a provisional peer entry for the IPV6 destination. The ICMPv6 packet will then be sent encapsulated in a UDP packet bound to the local server IPv4 address, and to the Teredo port. The rules of section 5.2.3 specify how the reception of this packet will be processed.

5.3 Teredo Server specification

The Teredo server is designed to be stateless. The Teredo server waits for incoming UDP packets at the Teredo Port, using the IPv4 address that has been selected for the service.

The Teredo server acts as an IPv6 router. As such, it will receive Router Solicitation messages, to which it will respond with Router Advertisement messages as explained in section 5.3.2; it may also receive other packets, for example ICMPv6 messages, which are processed according to the IPv6 specification.

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 and the UDP source port:

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

2) If the UDP packet is not a bubble or an ICMPv6 message, it should be discarded.

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

4) If the IPv6 source address is an IPv6 link-local address, the IPv6 destination address is the link-local scope all routers multicast address (FF02::2), and the packet contains an ICMPv6 Router Solicitation message, the packet SHOULD be accepted; it MUST be discarded if the server requires secure qualification and the authentication encapsulation is absent or cannot be verified.

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 SHOULD be accepted.

6) If the IPv6 source address is not a Teredo IPv6 address, and if the IPv6 destination address is a Teredo address allocated through this server, the packet SHOULD be accepted.

7) 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 the link-local scope all routers multicast address (FF02::2), or the link-local address of the server, the Teredo server processes the packet; it may have to process Router Solicitation messages and ICMPv6 Echo Request messages. If the destination IPv6 address is not a global scope IPv6 address, the packet MUST NOT be forwarded.

If the destination address is not a Teredo IPv6 address, the packet should be relayed to the IPv6 Internet using regular IPv6 routing.

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 server's IPv4 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 a Teredo client whose address is serviced by this specific server, the server should insert an origin indication in the first bytes of the UDP payload, as specified in section 5.1.1.

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 the origin indication encapsulation that will be sent with the response to the solicitation.

The Teredo server responds to the router solicitation by sending a Router Advertisement message [RFC2641]. The router advertisement MUST advertise the Teredo IPv6 prefix composed from the service prefix and the server's IPv4 address. The IPv6 source address should be set to a 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 message 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 server's IPv4 address and Teredo Port. If the cone bit of the client's IPv6 address is set to 1, the RA must be sent from a different IPv4 source address than the server address over which the RS was received; if the cone bit is set to zero, the response must be sent back from the same address.

Before sending the packet, the Teredo server 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 secure qualification is required, the server must insert a valid authentication parameter in the UDP packet carrying the router advertisement. The client identifier and the nonce value used in the authentication parameter must be the same identifier as received in the router solicitation; the confirmation byte should be set to zero if the client identifier is still valid, and a non-null value otherwise; the authentication value should be computed using the secret that corresponds to the client identifier.

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 should be able to receive packets sent over IPv4 and UDP by Teredo clients; they may apply filtering rules, e.g. only

accept packets from Teredo clients if they have previously sent traffic to these Teredo clients.

The receiving and sending rules used by Teredo relays are very similar to those of Teredo clients. Teredo relays must use a Teredo service port to transmit packets to Teredo clients; they must maintain a "list of peers", identical to the list of peers maintained by Teredo clients. However, Teredo relays do not have to perform the qualification procedure.

5.4.1 Transmission by relays to Teredo clients

When a Teredo relay has to transmit a packet to a Teredo client, it examines the destination IPv6 address. By definition, the Teredo relays will only send over UDP 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.

The relay then checks if there is an entry for this IPv6 address in the list of recent Teredo peers, and if the entry is still valid. The relay then performs the following:

1) If there is an entry for that IPv6 address in the list of peers, and if the status of the entry is set to "trusted", the IPv6 packet should be sent over UDP to the mapped IPv4 address and mapped UDP port of the entry. The client updates the date of last transmission in the peer entry.

2) If the destination is a Teredo IPv6 address in which the cone bit is set to 1, the packet is sent over UDP to the mapped IPv4 address and mapped UDP port extracted from that IPv6 address.

3) If the destination is a Teredo IPv6 address in which the cone bit is set to 0, the packet is queued. The Teredo relay creates a bubble whose source address is set to a local IPv6 address, and whose destination address is set to the Teredo IPv6 address of the packet's destination. The bubble is sent to the non-null server address corresponding to the Teredo destination. The packet will be de-queued and forwarded when a bubble or another packet will be received from this IPv6 address; if no such packet is received before a time-out of 2 seconds, the Teredo relay may repeat the bubble, up to three times.

In cases 2 and 3, the Teredo relay should create a peer entry for the IPv6 address; the entry status is marked as trusted in case 2 (cone NAT), not trusted in case 3. In case 3, if the Teredo relay happens to be located behind a non-cone NAT, it should also send a bubble directly to the mapped IPv4 address and mapped port number of the Teredo destination; this will "open the path" for the return

bubble from the Teredo client.

5.4.2 Reception from Teredo clients

The Teredo relay may receive packets from Teredo clients; the packets should normally only be sent by clients to which the relay previously transmitted packets, i.e. clients whose IPv6 address is present in the list of peers. Relays, like clients, use the packet reception procedure to maintain the date and time of the last interaction with the Teredo server, and the "list of recent peers."

When a UDP packet is received over the Teredo service port, the Teredo relay 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 relay examines whether the IPv6 source address is a valid Teredo address, and if the mapped IPv4 address and mapped port match the IPv4 source address and port number from which the packet is received. If this is not the case, the packet is silently discarded.

The Teredo relay then examines whether there is an entry for the IPv6 source address in the list of recent peers. If this is not the case, the packet may be silently discarded. If this is the case, the entry status is set to "trusted"; the relay updates the "date and time of the last interaction" to the current date and time.

Finally, the relay examines the destination IPv6 address. If the destination is the "all nodes multicast address", the packet should be processed locally. If the destination belongs to a range of IPv6 addresses served by the relay, the packet SHOULD be accepted, and forwarded to the destination. In the other cases, the packet SHOULD be silently discarded.

5.4.3 Difference between Teredo Relays and Teredo Servers

Because Teredo servers can relay Teredo packets over IPv6, all Teredo servers must be capable of behaving as Teredo relays. There is however no requirement that Teredo relays behave as Teredo servers.

The dual-role of server and relays implies an additional complexity for the programming of servers: the processing of incoming packets should be a combination of the server processing rules defined in 5.3.1, and the relay processing rules defined in 5.4.2.

5.5 Implementation of automatic sunset

Teredo is designed as an interim transition mechanism, and it is important that it should not be used any longer than necessary. The "sunset" procedure will be implemented by Teredo clients, servers

and relays, as specified in this section.

The Teredo-capable nodes MUST NOT behave as Teredo clients if they already have IPv6 connectivity through any other means, such as native IPv6 connectivity; in particular, nodes that have a global IPv4 address SHOULD obtain connectivity through the 6to4 service rather than through the Teredo service. The classic reason why a node that does not need connectivity would still enable the Teredo service is to guarantee good performance when interacting with Teredo clients; however, a Teredo-capable node that has IPv4 connectivity and that has obtained IPv6 connectivity outside the Teredo service MAY decide to behave as a Teredo relay, and still obtain good performance when interacting with Teredo clients.

The Teredo servers are expected to participate in the sunset procedure by announcing a date at which they will stop providing the service. This date depends on the availability of alternative solutions to their clients, such as "dual-mode" gateways that behave simultaneously as IPv4 NATs and IPv6 routers. Most Teredo servers will not be expected to operate more than a few years, perhaps until at most 2006.

Teredo relays are expected to have the same life span as Teredo servers.

6 Discussion of the solution

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

6.1 Why do we require address obfuscation?

The Teredo address, as specified in section 4.1.1, include an obfuscated copy of the mapped IPv4 address and UDP port of the client. This is done to prevent abusive NAT "smartness." We have experimental evidence that some NATs, probably in a desire to help applications operate more transparently across NATs, are programmed to look for occurrence of a 32-bit value that matches their local address, and to replace any such value by the local IP address allocated to the client; some may also attempt to translate the port values; this treatment is performed by some NATs even if they don't know the details of the application protocol. By obfuscating the address and port, we prevent the NAT from recognizing their own IPv4 address in the UDP packets exchanged between client and server, and avoid the errors caused by a possible rewriting.

6.2 Why do we have bubbles and lists of peers?

Our algorithm is designed to provide robustness: the client will always wait for a successful bubble or packet reception before transmitting data packets over UDP. This ensures that data packets will always be transmitted on a direct path to another Teredo

client, or on a direct path to the Teredo relay nearest from an IPv6 peer.

6.3 Why do servers only process bubbles and ICMPv6 messages?

In this specification, Teredo servers are requested to only forward some minimal packets: initial bubbles between Teredo clients, ICMPv6 messages between Teredo clients and IPv6 peers. This has two advantages: it greatly reduces the transmission load of servers, and it also helps in solving some the security issues.

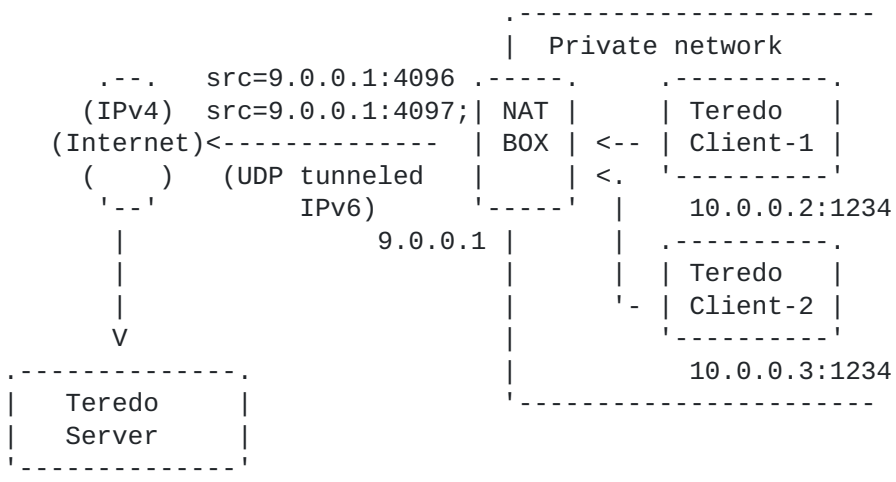
Restricting the traffic to a few bubbles means that the server will only have to carry a few hundred bits of data for any exchange between clients and peers. Previous designs allowed transmission of data through the server, which placed the server at risk: a lazily programmed client could skip sending bubbles, and send all its traffic through the server.

Restricting the server to only carry bubbles and ICMPv6 packets removes a privacy risk: if servers were allowed to carry data, a client could be convinced to send all its data through a rogue server, where it could easily be observed. With our design, a server does not see any actual data, and thus poses a much reduced privacy risk to its clients.

Restricting the type of packets that a server can relay also reduces another security risk, the use of the server as a reflection point in a denial of service attack. An attacker can induce a server to reflect packets towards a third party, but the structure of these packets is very limited, which prevents the use of the server in a "magic packet" attack.

6.4 What if two clients are behind the same NAT?

Our design choice implies some restrictions in the Teredo service. The first restriction concerns two clients connected to the Internet through the same NAT.



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. There is no guarantee that the NAT will be able to correctly process these packets; there is indeed some possibility that they may be lost, and that the two clients will not be able to communicate using Teredo.

We deliberately accept this restriction, as the alternative would be to relay the traffic between the two internal clients through the Teredo server. Since the clients are located behind the same NAT, in the same domain, there is a risk that the clients might exchange sensitive data without necessarily using proper protection; sending this data over the Internet to the Teredo server would expose the clients to a significant risk of information disclosure.

6.5 What about symmetric NAT?

The exchange of bubbles will fail if one of the Teredo clients is located behind a symmetric NAT; in practice, this means that clients located behind a symmetric NAT should not use Teredo.

6.6 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 an 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.**

6.7 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.

6.8 Scaling, failover and access control

The Teredo service is designed to impose minimal requirements on servers and relays: capability to send packets over IPv4 using a regular IPv4 address; capability to send and receive packets over IPv6; capability to advertise reachability of the Teredo service prefix in at least some limited scope. These minimal requirements make it easy to deploy a large number of servers and relays, thus ensuring scalability of the service.

Teredo clients may obtain a more resilient service if they can use several different servers. Teredo clients will detect that a server is failing through the failure of the qualification procedure; they may try at that point to obtain services from a different server.

6.9 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.

Implementations of Teredo should include an administrative control that explicitly enables use of the Teredo service; the service should not start if not explicitly authorized.

Implementations of Teredo should be configured to shut down the Teredo service when the Teredo client is connected within a "managed network", such as an enterprise network. For example, the implementation of Teredo in Microsoft Windows is configured to shut down the Teredo service if the client is a member of a Windows domain.

6.10 Why do we use the name Teredo?

"Teredo navalis" is the Latin name for a 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 North American harbors is a testimony to their newly retrieved cleanliness. The Teredo service should, in turn, contribute to a newly retrieved transparency of the Internet.

7 Use of Teredo to implement a 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. In fact, this can be done by configuring the client with:

- The IPv4 address of a Teredo server that acts as a tunnel broker
- A client identifier
- A shared secret with that server.

The Teredo client will use the secure qualification procedure, as specified in section 5.2.2. Instead of returning a Teredo prefix in the router advertisement, the server will return a globally routable IPv6 prefix; this prefix may be permanently assigned to the client, which would provide the client with a stable address. The server will have to keep state, i.e. memorize the association between the prefix assigned to the client and the mapped IPv4 address and mapped UDP port of the client.

The Teredo server will advertise reachability of the client prefix to the IPv6 Internet. Any packet bound to that prefix will be transmitted to the mapped IPv4 address and mapped UDP port of the client.

The Teredo client, when it receives the prefix, will notice that this prefix is a global IPv6 prefix, not in the form of a Teredo prefix. The client will at that point recognize that it should operate in tunnel mode. A client that operates in tunnel mode will execute a much simpler transmission procedure: it will forward any packet sent to the Teredo interface to the IPv4 address and Teredo UDP port of the server.

The Teredo client will have to perform the maintenance procedure described in section 5.2.5. The server will receive the router solicitation, and may notice a possible change of mapped IPv4 address and mapped UDP port that could result from the reconfiguration of the mappings inside the NAT. The server should continue advertising the same IPv6 prefix to the client, and should update the mapped IPv4 address and mapped UDP port associated to this prefix, if necessary.

8 Security Considerations

The main objective of Teredo is to provide nodes located behind a NAT with a globally routable IPv6 address. This enables such nodes to use IP security services such as IKE, AH or ESP. As such, we can argue that the service has a positive effect on network security. However, the security analysis must also envisage the negative effects of the Teredo services, which we can group in four categories: security risks of directly connecting a node to the IPv6 Internet, spoofing of Teredo servers to enable a man-in-the-middle attack, potential attacks aimed at denying the Teredo service to a Teredo client, and denial of service attacks against non-Teredo participating nodes that would be enabled by the Teredo service.

In the following, we review in detail these four types of issues, and we present mitigating strategies for each of them.

8.1 Opening a hole in the NAT

The very purpose of the Teredo service is to make a machine reachable through IPv6. By definition, the machine using the service will give up whatever "firewall" service was available in the NAT box; all services declared locally will become potential target of attacks from the entire IPv6 Internet. This may sound scary, but there are three mitigating factors.

The first mitigating factor is the possibility to restrict some services to only accept traffic from one of the limited address scopes defined in IPv6, e.g. link-local or site-local. There is no support for such scopes in Teredo, which implies that limited-scope services will not be accessed through Teredo, and will be restricted to whatever other IPv6 connectivity may be available, e.g. direct traffic with neighbors on the local link, behind the NAT.

The second mitigating factor is the possible use of a "local

firewall" solution, i.e. a piece of software that performs locally the kind of inspection and filtering that is otherwise performed in a perimeter firewall. Using such software is recommended.

The third mitigating factor, already noted, is the availability of end-to-end connectivity, which allows for deployment of IP security services such as IKE, AH or ESP. Using these services in conjunction with Teredo is a good policy, as it will protect the client from possible attacks in intermediate servers such as the NAT, the Teredo server, or the Teredo relay.

8.2 Using the Teredo service for a man-in-the-middle attack

The goal of the Teredo service is to provide hosts located behind a NAT with a globally reachable IPv6 address. There is a possible class of attacks against this service in which an attacker somehow intercepts the router solicitation, responds with a spoofed router advertisement, and provides a Teredo client with an incorrect address. The attacker may have one of two objectives: it may try to deny service to the Teredo client by providing it with an address that is in fact unreachable, or it may try to insert itself as a relay for all client communications, effectively enabling a variety of "man-in-the-middle" attack.

The secure qualification procedure described in section 5.2.2 enables a good protection against attacks in which a third party tries to spoof the server. To defeat this protection, the attacker could try to obtain a copy of the secret shared between client and server. The most likely way to obtain the shared secret is to listen to the traffic and mount an offline dictionary attack; to protect against this attack, the secret shared between client and server should be provisioned by an automatic procedure and contain sufficient entropy.

Another way to defeat the protection afforded by the signature procedure is to mount a complex attack, as follows:

1) Client prepares router solicitation, including authentication header.

2) Attacker intercepts the solicitation, and somehow manages to prevent it from reaching the server, for example by mounting a short duration DoS attack against the server.

3) Attacker replaces the source IPv4 address and source UDP port of the request by one of its own addresses and port, and forwards the modified request to the server.

4) Server dutifully notes the IPv4 address from which the packet is received, verifies that the Authentication encapsulation is correct, prepares a router advertisement, signs it, and sends it back to the incoming address, i.e. the attacker.

5) Attacker receives the advertisement, takes note of the mapping, replaces the IPv4 address and UDP port by the original values in the intercepted message, and sends the response to the client.

6) Client receives the advertisement, notes that the authentication header is present and is correct, and uses the proposed prefix and the mapped addresses in the origin indication encapsulation.

The root cause of the problem is that the NAT is, in itself, a man-in-the-middle attack. The Authentication encapsulation covers the encapsulated IPv6 packet, but does not cover the encapsulating IPv4 header and UDP header. It is very hard to devise an effective signature scheme, since the attacker does not do anything else than what the NAT legally does!

There are however several mitigating factors that lead us to avoid worrying too much about this attack. In practice, the gain from the attack is to either deny service to the client, or obtain a "man-in-the-middle" position; however, in order to mount the attack, the attacker must be able to suppress traffic originating from the client, i.e. have denial of service capability; the attacker must also be able to observe the traffic exchanged between client and inject its own traffic in the mix, i.e. have man-in-the-middle capacity. In summary, the attack is very hard to mount, and the gain for the attacker is minimal.

8.2.1 End-to-end security

The most effective line of defense of a Teredo client is probably not to try to secure the Teredo service itself: even if the mapping can be securely obtained, the attacker would still be able to listen to the traffic and send spoofed packets. Rather, the Teredo client should realize that, because it is located behind a NAT, it is in a situation of vulnerability; it should systematically try to encrypt its IPv6 traffic, using IPSEC. Even if the IPv4 and UDP headers are vulnerable, the use of IPSEC will effectively prevent spoofing and listening of the IPv6 packets by third parties. By providing each client with a global IPv6 address, Teredo enables the use of IPSEC and ultimately enhances the security of these clients.

8.3 Denial of the Teredo service

Our analysis outlines five ways to attack the Teredo service. There are counter-measures for each of these attacks.

8.3.1 Denial of service by a rogue relay

An attack can be mounted on the IPv6 side of the service by setting up a rogue relay, and letting that relay advertise a route to the Teredo IPv6 prefix. This is an attack against IPv6 routing, which can also be mitigated by the same kind of procedures used to

eliminate spurious route advertisements. Dual stack nodes that implement a "host local" Teredo relays are impervious to this attack.

8.3.1 Denial of service by server spoofing

In section 8.2, we discussed the use of spoofed router advertisements to insert an attacker in the middle of a Teredo conversation. The spoofed router advertisements can also be used to provision a client with an incorrect address, pointing to either a non existing IPv4 address or to the IPv4 address of a third party.

The Teredo client will detect the attack when it fails to receive traffic through the newly acquired IPv6 address. The attack can be mitigated by using the authentication encapsulation.

8.3.2 Denial of service by exceeding the number of peers

A Teredo client manages a cache of recently-used peers, which makes it stateful. It is possible to mount an attack against the client by provoking it to respond to packets that appear to come from a large number of Teredo peers, thus trashing the cache and effectively denying the use of direct communication between peers. The effect will only last as long as the attack is sustained.

8.3.3 Attacks against the local discovery procedure

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 5.2.8 mitigate this attack. Clients who are more interested in security than in performance could decide to disable the local discovery procedure; however, if local discovery is disabled, traffic between local nodes will end up being relayed through a server external to the local network, which has questionable security implications.

8.3.4 Attacking the Teredo servers and relays

It is possible to mount a brute force denial of service attack against the Teredo servers by sending them a very large number of packets. This attack will have to be "brute force", since the servers are stateless, and can be designed to process all the packets that are sent on their access line.

The brute force attack against the Teredo servers is mitigated if clients are ready to "failover" to another server. Bringing down the servers will however force the clients that change servers to renumber their Teredo address.

It is also possible to mount a brute force attack against a Teredo relay. This attack is mitigated if the relay under attack stops

announcing the reachability of the Teredo service prefix to the IPv6 network: the traffic will be picked up by the next relay.

8.4 Denial of service against non-Teredo nodes

There is a widely expressed concern that transition mechanisms such as Teredo can be used to mount denial of service attacks, by injecting traffic at locations where it is not expected. These attacks fall in three categories: using the Teredo servers as a reflector in a denial of service attack, using the Teredo server to carry a denial of service attack against IPv6 nodes, and using the Teredo relays to carry a denial of service attack against IPv4 nodes. The analysis of these attacks follows. A common mitigating factor in all cases is the "regularity" of the Teredo traffic, which contains highly specific patterns such as the Teredo UDP port, or the Teredo IPv6 prefix. In case of attacks, these patterns can be used to quickly install filters and remove the offending traffic.

8.4.1 Laundering DOS attacks from IPv4 to IPv4

An attacker can use the Teredo servers as reflectors in a denial of service attack aimed at an IPv4 target. The attacker can do this in one of two ways. The first way is to construct a Router Solicitation message and post it to a Teredo server, using as IPv4 source address the spoofed address of the target; the Teredo server will then send a Router advertisement message to the target. The second way is to construct a Teredo IPv6 address using the Teredo prefix, the address of a selected server, the IPv4 of the target, and an arbitrary UDP port, and to then send packets bound to that address to the selected Teredo server.

Reflector attacks are discussed in [REFLECT], which outlines various mitigating techniques against such attacks. One of these mitigations is to observe that 'the traffic generated by the reflectors [has] sufficient regularity and semantics that it can be filtered out near the victim without the filtering itself constituting a denial-of-service to the victim ("collateral damage").' The traffic reflected by the Teredo servers meets this condition: it is clearly recognizable, since it originates from the Teredo UDP port; it can be filtered out safely if the target itself is not a Teredo user. In addition, the packets relayed by servers will carry an Origin indication encapsulation, which will help determining the source of the attack.

8.4.2 DOS attacks from IPv4 to IPv6

An attacker may use the Teredo servers to launch a denial of service attack against an arbitrary IPv6 destination. The attacker will build an IPv6 packet bound for the target, and will send that packet to the IPv4 address and UDP port of a Teredo server, to be relayed from there to the target over IPv6.

The address checks specified in section 5.3.1 provide some protection against this attack, as they ensure that the IPv6 source address will be consistent with the IPv4 source address and UDP source port used by the attacker: if the attacker cannot spoof the IPv4 source address, the target will be able to determine the origin of the attack.

The address checks ensure that the IPv6 source address of packets forwarded by servers will start with the IPv6 Teredo prefix. This is a mitigating factor, as sites under attack could use this to filter out all packets sourced from that prefix during an attack. This will result in a partial loss of service, as the target will not be able to communicate with legitimate Teredo hosts that use the same prefix; however, the communication with other IPv6 hosts will remain unaffected, and the communication with Teredo hosts will be able to resume when the attack has ceased.

The ICMP Traceback (ITRACE) working group is considering systems for "tracing" the source of DOS attacks. According to the proposal, when forwarding packets, routers can, with a low probability, generate a Traceback message that is sent along to the destination; with enough Traceback messages from enough routers along the path, the traffic source and path can be determined. This set up assumes that the source and destination are both using the same version of IP. In the Teredo case, the ICMP Traceback packets will be sent to the Teredo server, not the final destination. It is conceivable to "map" the IPv4 traceback to an IPv6 traceback sent by the Teredo server; the details of the solution should be specified by the ITRACE working group.

8.4.3 DOS attacks from IPv6 to IPv4

An attacker with IPv6 connectivity may use the Teredo relays to launch a denial of service attack against an arbitrary IPv4 destination. The attacker will compose a Teredo IPv6 address using the Teredo prefix, a null server address, the IPv4 address of the target, an arbitrary UDP port, and an arbitrary node identifier. The attacker will send IPv6 packets to that address; the packets will be routed to the nearest Teredo relay, and forwarded from there to the target.

The address checks specified in 5.4 are limited to verifying that packets are only relayed to a global IPv4 address. This rules out a class of attack in which the packets are bound to a broadcast or multicast address. It also rules out another class of attack in which the packets are bound for a private IPv4 address that would be reachable by the relay.

The attack can be targeted at arbitrary UDP ports, such as for example the DNS port of a server. The UDP payload must be a well-formed IPv6 packet, and is thus unlikely to be accepted by any well-written UDP service; in most case, the only effect of the attack

will be to overload the target with random traffic.

A special case occurs if the attack is directed to an echo service. The service will echo the packets. Since the echo service sees the request coming from the IPv4 address of the relay, the echo replies will be sent back to the same relay. According to the rules specified in 5.4, these packets will be discarded by the Teredo relay. This is not a very efficient attack against the Teredo relays - establishing a legitimate session with an actual Teredo host would create more traffic.

The IPv6 packets sent to the target contain the IPv6 address used by the attacker. If ingress filtering is used in the IPv6 network, this address will be hard to spoof. If ingress filtering is not used, the attacker can be traced if the IPv6 routers use a mechanism similar to ICMP Traceback. The ICMP messages will normally be collected by the same relays that forward the traffic from the attacker; the relays can use these messages to identify the source of an ongoing attack. The details of this solution should be specified by the ITRACE working group.

9 IANA Considerations

This memo documents a request to IANA to allocate a Teredo IPv6 service prefix.

10 Copyright

The following copyright notice is copied from RFC 2026 [Bradner, 1996], Section 10.4, and describes the applicable copyright for this document.

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 other 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 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, Mohit Talwar, Joseph Davies 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. 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, Thomas Narten, Anssi Porttikivi, Pekka Savola, and Eng Soo Guan.

13 References

[RFC768] J. Postel, "User Datagram Protocol", RFC 768, August 1980.

[RFC791] J. Postel, "Internet Protocol", RFC 791, September 1981.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC1918] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, "Address Allocation for Private Internets", RFC 1918, February 1996.

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC2460] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2461] T. Narten, E. Nordmark, W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[RFC2462] T. Narten, S. Thomson, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

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

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

[RFC1750] D. Eastlake, S. Crocker, J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[SYNCHRO] S. Floyd, V. Jacobson, "The synchronization of periodic routing messages", ACM SIGCOMM'93 Symposium, September 1993.

[REFLECT] V. Paxson, "An analysis of using reflectors for distributed denial of service attacks." Computer Communication Review, ACM SIGCOMM, Volume 31, Number 3, July 2001, pp 38-47.

14 Authors' Addresses

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

Email: huitema@microsoft.com

Table of Contents: