Internet Engineering Task Force (IETF)
Request for Comments: 6081
Updates: 4380
Category: Standards Track
ISSN: 2070-1721

D. Thaler
Microsoft

January 2011

Teredo Extensions

Abstract

This document specifies a set of extensions to the Teredo protocol. These extensions provide additional capabilities to Teredo, including support for more types of Network Address Translations (NATs) and support for more efficient communication.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6081.

Table of Contents

1. Introduction

This document specifies extensions to the Teredo protocol, as specified in [RFC4380]. These extensions provide additional capabilities to Teredo, including support for more types of Network Address Translations (NATs) and support for more efficient communication.

2. Terminology

Because this document extends [RFC4380], it uses the following terminology, for consistency with [RFC4380].

Address-Restricted NAT: A restricted NAT that accepts packets from an external host's IP address X and port Y if the internal host has sent a packet that is destined to IP address X regardless of the destination port. In the terminology of [RFC4787], this is a NAT with Endpoint-Independent Mapping and Address-Dependent Filtering.

Address-Symmetric NAT: A symmetric NAT that has multiple external IP addresses and that assigns different IP addresses and ports when communicating with different external hosts.

Cone NAT: A NAT that maps all requests from the same internal IP address and port to the same external IP address and port. Furthermore, any external host can send a packet to the internal host by sending a packet to the mapped external address and port. In the terminology of [RFC4787], this is a NAT with Endpoint-Independent Mapping and Endpoint-Independent Filtering.

Direct Bubble: A Teredo bubble that is sent directly to the IPv4 node whose Teredo address is contained in the Destination field of the IPv6 header, as specified in Section 2.8 of [RFC4380]. The IPv4 Destination Address and UDP Destination Port fields contain a mapped address/port.

Echo Test: A mechanism to predict the mapped address/port a sequential port-symmetric NAT is using for a client behind it.

Hairpinning: A feature that is available in some NATs where two or more hosts are positioned behind a NAT and each of those hosts is assigned a specific external (public) address and port by the NAT. Hairpinning support in a NAT allows these hosts to send a packet to the external address and port that is assigned to one of the other hosts, and the NAT automatically routes the packet back to the correct host. The term hairpinning is derived from the behavior of the packet, which arrives on, and is sent out to, the same NAT interface.

Indirect Bubble: A Teredo bubble that is sent indirectly (via the destination's Teredo server) to another Teredo client, as specified in Section 5.2.4 of [RFC4380].

Local Address/Port: The IPv4 address and UDP port from which a Teredo client sends Teredo packets. The local port is referred to as the Teredo service port in [RFC4380]. The local address of a node may or may not be globally routable because the node can be located behind one or more NATs.

Mapped Address/Port: A global IPv4 address and a UDP port that results from the translation of a node's own local address/port by one or more NATs. The node learns these values through the Teredo protocol as specified in [RFC4380]. For symmetric NATs, the mapped address/port can be different for every peer with which a node tries to communicate.

Network Address Translation (NAT): The process of converting between IP addresses used within an intranet or other private network and Internet IP addresses.

Nonce: A time-variant random value used in the connection setup phase to prevent message replay and other types of attacks.

Peer: A Teredo client with which another Teredo client needs to communicate.

Port-Preserving NAT: A NAT that translates a local address/port to a mapped address/port such that the mapped port has the same value as the local port, as long as that same mapped address/port has not already been used for a different local address/port.

Port-Restricted NAT: A restricted NAT that accepts packets from an external host's IP address X and port Y only if the internal host has sent a packet destined to IP address X and port Y. In the terminology of [RFC4787], this is a NAT with Endpoint-Independent Mapping and Address and Port-Dependent Filtering.

Port-Symmetric NAT: A symmetric NAT that has only a single external IP address and hence only assigns different ports when communicating with different external hosts.

Private Address: An IPv4 address that is not globally routable but is part of the private address space specified in Section 3 of [RFC1918].

Public Address: An external global address used by a NAT.

Restricted NAT: A NAT where all requests from the same internal IP address and port are mapped to the same external IP address and port. Unlike the cone NAT, an external host can send packets to an internal host (by sending a packet to the external mapped address and port) only if the internal host has first sent a packet to the external host. There are two kinds of restricted NATs: address-restricted NATs and port-restricted NATs.

Sequential Port-Symmetric NAT: A port-symmetric NAT that allocates external ports sequentially for every {internal IP address and port, destination IP address and port} tuple. The delta used in the sequential assignment is typically 1 or 2 for most such NATs.

Symmetric NAT: A NAT where all requests from the same internal IP address and port and to the same destination IP address and port are mapped to the same external IP address and port. Requests from the same internal IP address and port to a different destination IP address and port may be mapped to a different external IP address and port. Furthermore, a symmetric NAT accepts packets received from an external host's IP address X and port Y only if some internal host has sent packets to IP address X and port Y. In the terminology of [RFC4787], this is a NAT with a mapping behavior of either Address-Dependent Mapping or Address- and Port-Dependent Mapping, and a filtering behavior of either Address-Dependent Filtering or Address- and Port-Dependent Filtering.

Teredo Bubble: A Teredo control message (specified in Section 2.8 of [RFC4380]) that is used to create a mapping in a NAT. There are two types of Teredo bubbles: direct bubbles and indirect bubbles.

Teredo Client: A node that has access to the IPv4 Internet and wants to gain access to the IPv6 Internet using the Teredo protocol.

Teredo IPv6 Address: An IPv6 address of a Teredo client, as specified in Section 2.14 of [RFC4380].

Teredo Secondary Server Address: A secondary IPv4 address of a Teredo server with which a Teredo client is configured, as specified in Section 5.2 of [RFC4380].

Teredo Server: A node that has a globally routable address on the IPv4 Internet, and is used as a helper to provide IPv6 connectivity to Teredo clients.

Teredo Server Address: A (primary) IPv4 address of a Teredo server with which a Teredo client is configured, as specified in Section 5.2 of [RFC4380].

UPnP-enabled NAT: A NAT that has the UPnP device control protocol enabled, as specified in [UPNPWANIP]. (Note that today, by default, most UPnP-capable NATs have the UPnP device control protocol disabled.)

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 RFC 2119 [RFC2119].

3. Overview

The Teredo protocol (as specified in [RFC4380]) enables nodes located behind one or more IPv4 NATs to obtain IPv6 connectivity by tunneling packets over UDP.

When a node behind a NAT needs to communicate with a peer (i.e., another node) that is behind a NAT, there are four sets of IPv4 address/port pairs of interest:

- The node's own IPv4 address/port.
- The external IPv4 address/port to which the node's NAT translates.
- The peer's own IPv4 address/port.
- The external IPv4 address/port to which the peer's NAT translates.

When the node sends a packet to a peer, the node needs to send it from the node's own IPv4 address/port, destined to the peer's external IPv4 address/port. By the time it arrives at the peer (i.e., after passing through both NATs), the peer will see the same packet as coming from the node's external IPv4 address/port, destined to the peer's own IPv4 address/port.

In this document, the term local address/port refers to a Teredo client's own IPv4 address/port, and mapped address/port refers to the external IPv4 address/port to which its NAT translates the local address/port. That is, the mapped address/port is what the IPv4 Internet sees the Teredo client as.

A Teredo client running on a node communicates with a Teredo server to discover its mapped address/port. The mapped address/port, along with the Teredo server address, is used to generate an IPv6 address known as a Teredo IPv6 address. This allows any peer that gets the node's IPv6 address to easily determine the external IPv4 address/port to which to send IPv6 packets encapsulated in IPv4 UDP messages.

This document specifies extensions to the Teredo protocol. These Teredo extensions are independent of each other and can be implemented in isolation, except that the UPnP-Symmetric NAT Extension and the Port-Preserving Symmetric NAT Extension both require the Symmetric NAT Support Extension to be implemented. An implementation of this specification can support any combination of the Teredo extensions, subject to the above-mentioned restriction.

The following matrix outlines the connectivity improvements of some of the extensions outlined in this document.

	Destination NAT								
Source NAT	Cone	Addr. rest.	Port rest.	UPnP Port rest.	UPnP Port symm.	Port-pres. Port-symm.	Seq. Port-symm.	Port-symm.	Addr symm
Cone	Yes	Yes	Yes	Yes	SNS	SNS	SNS	SNS	SNS
Address restricted	Yes	Yes	Yes	Yes	SNS	SNS	SNS	SNS	No
Port restricted	Yes	Yes	Yes	Yes	No	SNS+ PP	SNS+ SS	No	No
UPnP Port-restricted	Yes	Yes	Yes	Yes	SNS+ UPnP	No	No	No	No
UPnP Port symmetric	SNS	SNS	No	SNS+ UPnP	SNS+ UPnP	No	No	No	No
Port-preserving Port-symmetric	SNS	SNS	SNS + PP	No	No	SNS + PP	SNS + SS	No	No
Sequential Port-symmetric	SNS	SNS	SNS + SS	No	No	No	No	No	No
Port-symmetric	SNS	SNS	No	No	No	No	No	No	No
Address-symmetric	SNS	No	No	No	No	No	No	No	No

Yes = Supported by [RFC4380].

SNS = Supported with the Symmetric NAT Support Extension.

SNS+UPnP = Supported with the Symmetric NAT Support Extension and UPnP Symmetric NAT Extension.

SNS+PP = Supported with the Symmetric NAT Support Extension and Port-Preserving Symmetric NAT Extension.

SNS+SS = Supported with the Symmetric NAT Support Extension and Sequential Port-Symmetric NAT Extension.

No = No connectivity.

Figure 1: Matrix of Connectivity Improvements for Teredo Extensions

Note that as with [RFC4380], if the qualification process is not successful, Teredo will not be configured with an IPv6 address, and connectivity will function as if Teredo were not present. Similarly, for any combination of NAT types that are not supported by Teredo and the extensions defined herein, the connectivity tests between a client and a peer will fail within a finite period of time, allowing the client to handle this case as with any other type of unreachable destination address (e.g., by trying another address of the destination such as a native IPv4 address).

3.1. Symmetric NAT Support Extension

The qualification procedure (as specified in Section 5.2.1 of [RFC4380]) is a process that allows a Teredo client to determine the type of NAT that it is behind, in addition to its mapped address/port as seen by its Teredo server. However, Section 5.2.1 of [RFC4380] suggests that if the client learns it is behind a symmetric NAT, the Teredo client should go into an "offline state" where it is not able to use Teredo. The primary reason for doing so is that it is not easy for Teredo clients to connect to each other if either or both of them are positioned behind a symmetric NAT. Because of the way a symmetric NAT works, a peer sees a different mapped address/port in the IPv4/UDP headers of packets coming from a Teredo client than the node's Teredo server sees (and hence appears in the node's Teredo IPv6 address). Consequently, a symmetric NAT does not allow incoming packets from a peer that are addressed to the mapped address/port embedded in the node's Teredo IPv6 address. Thus, the incoming packets are dropped and communication with Teredo clients behind symmetric NATs is not established.

With the Symmetric NAT Support Extension, Teredo clients begin to use Teredo even after they detect that they are positioned behind a symmetric NAT.

Consider the topology shown in Figure 2. Teredo Client B uses Teredo Server 2 to learn that its mapped address/port is 192.0.2.10:8192, and constructs a Teredo IPv6 address, as specified in Section 4 of [RFC4380]. Hence, c633:6476 is the hexadecimal value of the address of Teredo Server 2 (198.51.100.118), the mapped port is exclusive-OR'ed with 0xffff to form dfff, and the Mapped Address is exclusive-OR'ed with 0xffffffff to form 3fff:fdf5.

Teredo Client A uses Teredo Server 1 to learn that its mapped address/port is 192.0.2.1:4096 and, with this extension, constructs a Teredo IPv6 address (as specified in Section 4 of [RFC4380]) even though it learns that it is behind a symmetric NAT. Hence, cb00:7178 is the hexadecimal value of the address of Teredo Server 1 (203.0.113.120), the mapped port is exclusive-OR'ed with 0xffff to form efff, and the Mapped Address is exclusive-OR'ed with 0xffffffff to form 3fff:fdfe.

The Symmetric NAT Support Extension enables a Teredo client positioned behind a symmetric NAT to communicate with Teredo peers positioned behind a cone or address-restricted NATs as follows, depending on what side initiates the communication.

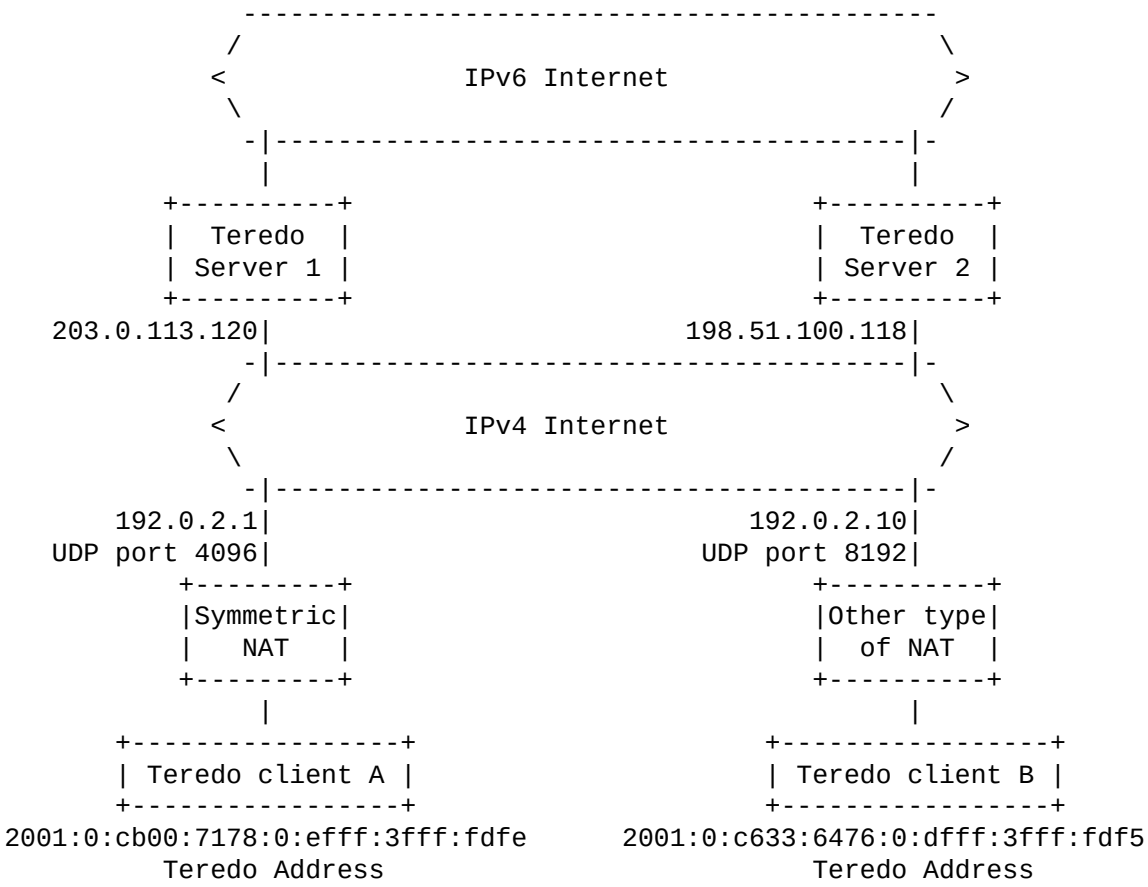


Figure 2: Symmetric NAT Example

In the first case, assume that a Teredo Client B (B) positioned behind a cone or address-restricted NATs initiates communication with Teredo Client A (A) positioned behind a symmetric NAT. B sends an

indirect bubble via A's server (Teredo Server 1) to A, and A responds with a direct bubble. This direct bubble reaches B, because it is positioned behind a cone or address-restricted NAT. However, the mapped address/port in the IPv4/UDP headers of the direct bubble are different from the mapped address/port embedded in A's Teredo IPv6 address. B therefore remembers the mapped address/port of the direct bubble and uses them for future communication with A, and thus communication is established.

In the second case, assume that A, positioned behind a symmetric NAT, initiates communication with B, positioned behind a cone or address-restricted NAT. A sends an indirect bubble to B via B's server (Teredo Server 2), and B responds with a direct bubble. This direct bubble is dropped by A's symmetric NAT because the direct bubble is addressed to the mapped address/port embedded in A's Teredo IPv6 address. However, communication can be established by having B respond with an indirect bubble via A's server (Teredo Server 1). Now the scenario is similar to the first case and communication will be established.

3.2. UPnP-Enabled Symmetric NAT Extension

The UPnP-enabled Symmetric NAT Extension is dependent on the Symmetric NAT Support Extension. Only if Teredo clients have been enabled to acquire a Teredo IPv6 address in spite of being behind a symmetric NAT will this extension help in traversing UPnP-enabled Symmetric NATs.

The Symmetric NAT Support Extension enables communication between Teredo clients behind symmetric NATs with Teredo clients behind cone NATs or address-restricted NATs. However, clients behind symmetric NATs can still not communicate with clients behind port-restricted NATs or symmetric NATs.

Referring again to Figure 2 (see Section 3.1), assume that Teredo Client A is positioned behind a symmetric NAT and initiates communication with Client B, which is positioned behind a port-restricted NAT. Client A sends a direct bubble and an indirect bubble to Client B via Client B's server (Teredo Server 2). As per the characteristics of the symmetric NAT, the IPv4 source of the direct bubble contains a different mapped address and/or port than the one embedded in the Teredo server. This direct bubble is dropped because Client B's NAT does not have state to let it pass through, and Client B does not learn the mapped address/port used in the IPv4/UDP headers. In response to the indirect bubble from Client A, Client B sends a direct bubble destined to the mapped address/port embedded in Client A's Teredo IPv6 address. This direct bubble is dropped because Client A's NAT does not have state to accept packets

destined to that mapped address/port. The direct bubble does, however, cause Client B's NAT to set up outgoing state for the mapped address/port embedded in Client A's Teredo IPv6 address.

As described in Section 3.1, Client B also sends an indirect bubble that elicits a direct bubble from Client A. Unlike the case in Section 3.1, however, the direct bubble from Client A is dropped as Client B's NAT does not have state for the mapped address/port that Client A's NAT uses. Note that Client B's NAT is port-restricted and hence requires both the mapped address and port to be the same as in its outgoing state, whereas in Section 3.1, Client A's NAT was a cone or address-restricted NAT which only required the mapped address (but not port) to be the same. Thus, communication between Client A and Client B fails. If Client B were behind a symmetric NAT, the problem is further complicated by Client B's NAT using a different outgoing mapped address/port than the one embedded in Client B's Teredo IPv6 address.

If a Teredo client is separated from the global Internet by a single UPnP-enabled symmetric or port-restricted NAT, it can communicate with other Teredo clients that are positioned behind a single UPnP-enabled symmetric or port-restricted NAT as follows.

Teredo clients, before communicating with the Teredo server during the qualification procedure, use UPnP to reserve a translation from a local address/port to a mapped-address/port. Therefore, during the qualification procedure, the Teredo server reflects back the reserved mapped address/port, which then is included in the Teredo IPv6 address. The mapping created by UPnP allows the NAT to forward packets destined for the mapped address/port to the local address/port, independent of the source of the packets. It typically does not, however, cause packets sourced from the local address/port to be translated to have the mapped address/port as the external source and hence continues to function as a symmetric NAT in this respect.

Thus, a Teredo client, positioned behind a UPnP-enabled symmetric NAT, can receive a direct bubble sent by any Teredo peer. The Teredo client compares the peer's mapped address/port as seen in the IPv4/UDP headers with the mapped address/port in the peer's Teredo IPv6 address. If the two mappings are different, the packet was sent by another Teredo client positioned behind a symmetric NAT. The Symmetric NAT Support Extension suggested that the Teredo client use the peer's mapped address/port seen in the IPv4/UDP headers for future communication. However, because symmetric NAT-to-symmetric NAT communication would not have been possible anyway, the Teredo client sends back a direct bubble to the mapped port/address embedded

in the peer's Teredo IPv6 address. If the peer is also situated behind a UPnP-enabled NAT, the direct bubble will make it through and communication will be established.

Even though communication is established between the two Teredo IPv6 addresses, the mappings will be asymmetric in the two directions of data transfer. Specifically, incoming packets will be destined to the reserved mapped address/port that is embedded in the Teredo IPv6 address. Outgoing packets will instead appear to come from a different mapped address/port due to the symmetric NAT behavior.

3.3. Port-Preserving Symmetric NAT Extension

The Port-Preserving Symmetric NAT Extension is dependent on the Symmetric NAT Support Extension (Section 3.1). Only if Teredo clients have been enabled to acquire a Teredo IPv6 address in spite of being behind a symmetric NAT will this extension help in traversing port-preserving symmetric NATs.

The Symmetric NAT Support Extension enables communication between Teredo clients behind symmetric NATs with Teredo clients behind cone NATs or address-restricted NATs. However, clients behind symmetric NATs can still not communicate with clients behind port-restricted or symmetric NATs, as described in Section 3.2. Note that the Port-Preserving Symmetric NAT Extension described here is independent of the UPnP-enabled Symmetric NAT Extension, described in Section 3.2.

If a Teredo client is positioned behind a port-preserving symmetric NAT, the client can communicate with other Teredo clients positioned behind a port-restricted NAT or a port-preserving symmetric NAT as follows.

Teredo clients compare the mapped port learned during the qualification procedure with their local port to determine if they are positioned behind a port-preserving NAT. If both the mapped port and the local port have the same value, the Teredo client is positioned behind a port-preserving NAT. At the end of the qualification procedure, the Teredo client also knows if it is positioned behind a symmetric NAT, as described in Section 3.1.

Teredo clients positioned behind port-preserving symmetric NATs can also listen on randomly chosen local ports. If the randomly chosen local port has not been used by the symmetric NAT as a mapped port in a prior port-mapping, the NAT uses the same port number as the mapped port. Thus, the challenge is to get the first direct bubble sent out from the random port to be destined to a valid destination address and port. When the mapped address/port is embedded in the destination's Teredo IPv6 address, this is easy.

The communication setup is more complicated when the destination Teredo client is also positioned behind a port-preserving symmetric NAT. In such a case, both Teredo clients need to send their first direct bubbles to the correct destination mapped address/port. Thus, the protocol messages, which communicate one Teredo client's random port number to the other Teredo client, must be exchanged indirectly (via Teredo servers). When one Teredo client has access to the other Teredo client's random port number, it can send a direct bubble destined to the mapped address embedded in the destination's Teredo IPv6 address, and the mapped port can be the same as the destination's random port number. If both NATs are port-preserving, port-preserved mappings are created on both NATs and the second direct bubble succeeds in reaching the destination.

3.4. Sequential Port-Symmetric NAT Extension

The Sequential Port-Symmetric NAT Extension is dependent on the Symmetric NAT Support Extension (Section 3.1). This extension helps in traversing a sequential port-symmetric NAT only if Teredo clients are enabled to acquire a Teredo IPv6 address even when behind a symmetric NAT.

When the Sequential Port-Symmetric NAT Extension is used, if a Teredo client is positioned behind a sequential port-symmetric NAT, the client can communicate with other Teredo clients that are positioned behind a port-restricted NAT as follows.

During qualification, if the client discovers it is behind a symmetric NAT that is not port-preserving, the client assumes by default that it is behind a sequential port-symmetric NAT. This assumption is proactive for the following reasons:

- There is no perfect method of discovering whether the client is behind a sequential port-symmetric NAT.
- These kinds of NATs are notorious for changing their behavior. At times, they could be sequential port-symmetric and at other times not.
- There is no other solution for symmetric NAT traversal so this is a last resort.

Teredo clients positioned behind sequential port-symmetric NATs can also listen on a randomly chosen local port when communicating with a peer. To predict the external port being used for a given peer, the client sends three packets:

- Packet 1 is a router solicitation (as specified in Section 5.2.1 of [RFC4380]) sent to the Teredo server address.

- Packet 2 is a direct bubble sent to the peer.

- Packet 3 is a router solicitation sent to the secondary Teredo server address.

As part of the normal Teredo protocol, the Teredo server responds to packets 1 and 3. Based on the information in the responses, the client now knows that Packet 1 was seen as coming from one external port, and Packet 3 was seen as coming from another external port. Assuming the NAT is a sequential port-symmetric NAT, the external port for Packet 2 is estimated (or predicted) to be midway between the external ports for Packets 1 and 3. Note that because other applications might also have been using the NAT between packets 1 and 3, the actual port might not be exactly the midpoint.

The Teredo client then communicates the predicted port to its peer, which sends a direct bubble to the communicated port. If the communicated port is indeed the external port for Packet 2, the direct bubble will reach the Teredo client.

3.5. Hairpinning Extension

Hairpinning support in a NAT routes packets that are sent from a private (local) address destined to a public (mapped) address of the NAT, back to another private (local) destination address behind the same NAT. If hairpinning support is not available in a NAT, two Teredo clients behind the same NAT are not able to communicate with each other, as specified in Section 8.3 of [RFC4380].

The Hairpinning Extension enables two clients behind the same NAT to talk to each other when the NAT does not support hairpinning. This process is illustrated in the following diagram.

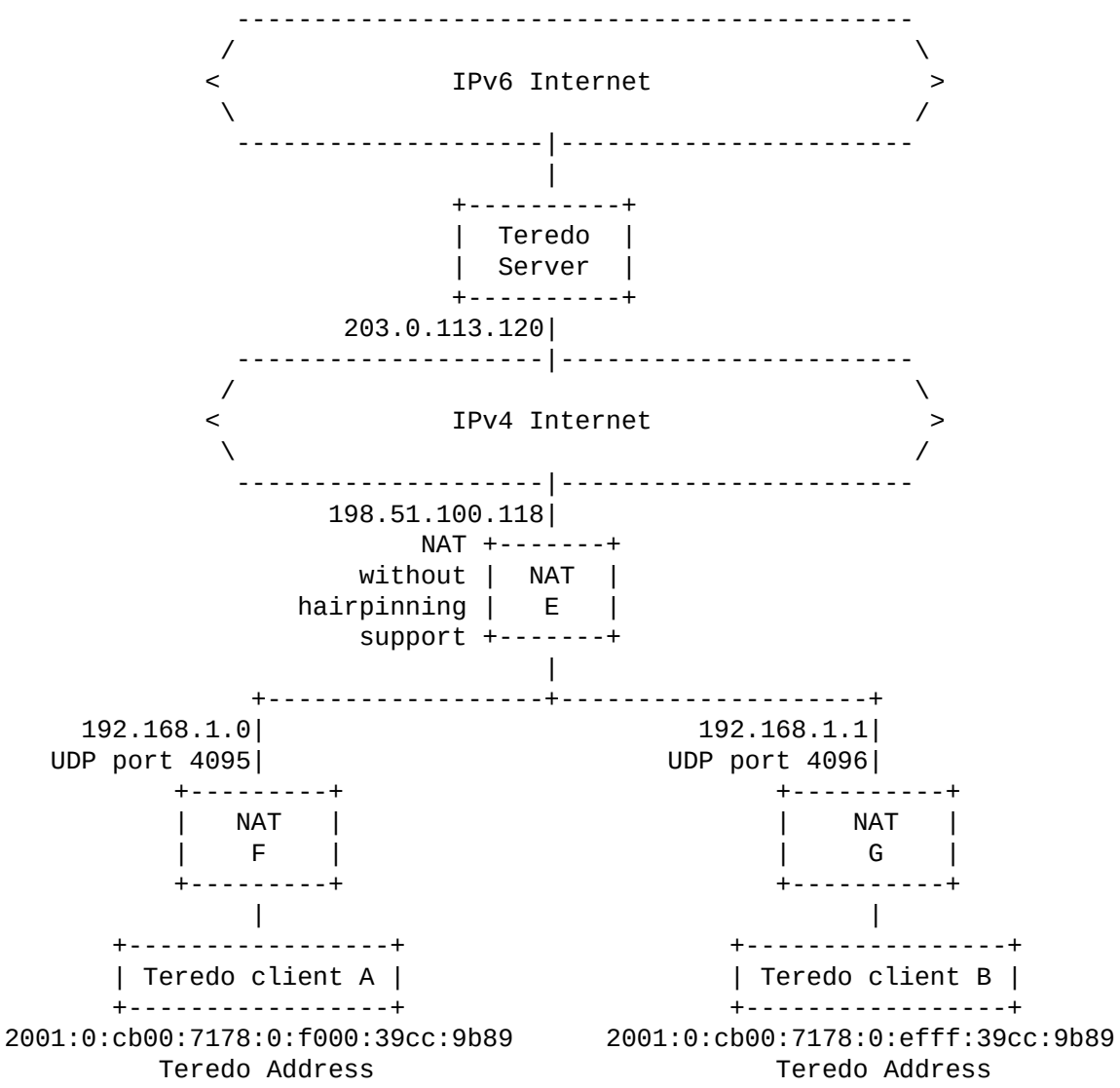


Figure 3: Hairpinning Example

The Teredo Client A (A) includes, as part of its indirect bubble sent to Teredo Client B (B), its local address/port. B, upon receiving the indirect bubble, tries to establish communication by sending direct bubbles to the mapped address/port of A, and also to the local address/port of B.

If a Teredo client is part of a multi-NAT hierarchy and the NAT to which the Teredo client is connected supports the UPnP protocol (as specified in [UPNPWANIP]), the Teredo client can use UPnP to determine the mapped address/port assigned to it by the NAT. This

information can be included along with the local address/port when sending the indirect bubble. The destination Teredo client now tries to establish a connection by sending direct bubbles to the mapped address/port in the Teredo IPv6 address, to the local address/port included in the bubble, and also to the mapped address/port included in the bubble.

Note that UPnP support is only required if the Teredo clients are behind different NATs in a multi-NAT hierarchy. Without UPnP support, the Hairpinning Extension still allows two hosts behind the same non-hairpinning NAT to communicate using their Teredo IPv6 addresses.

3.6. Server Load Reduction Extension

If communication between a Teredo client and a Teredo peer was successfully established but at a later stage was silent for a while, for efficiency, it is best to refresh the mapping state in the NATs that are positioned between them. To refresh the communication between itself and a Teredo peer, a Teredo client needs to solicit a direct bubble response from the Teredo peer. An indirect bubble is sent to solicit a direct bubble response from a Teredo peer, as specified in Section 5.2.4 of [RFC4380]. However, these indirect bubbles increase the load on the Teredo server.

The Server Load Reduction Extension allows Teredo clients to send direct bubbles most of the time instead of sending indirect bubbles all of the time in the following way:

1. When a Teredo client tries to refresh its communication with a Teredo peer, it uses a direct bubble instead of an indirect bubble. However, because direct bubbles do not normally solicit a response, the direct bubble format is extended to be able to solicit a response.

2. When a Teredo client receives a direct bubble that is soliciting a response, the Teredo client responds with a direct bubble.

3. If attempts to re-establish communication with the help of direct bubbles fail, the Teredo client starts over the process of establishing communication with the Teredo peer, as specified in Section 5.2.4 of [RFC4380].

4. Message Syntax

All Teredo messages are transported over the User Datagram Protocol (UDP), as specified in Section 3 of [RFC4380].

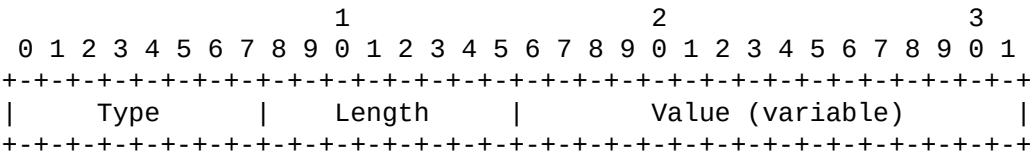
In addition, Section 5.2.3 of [RFC4380] states:

> An IPv6 packet is deemed valid if it conforms to [RFC2460]: the protocol identifier should indicate an IPv6 packet and the payload length should be consistent with the length of the UDP datagram in which the packet is encapsulated. In addition, the client should check that the IPv6 destination address correspond [sic] to its own Teredo address.

This document updates the word "consistent" above as follows. The IPv6 payload length is "consistent" with the length of the UDP datagram if the IPv6 packet length (i.e., the Payload Length value in the IPv6 header plus the IPv6 header size) is less than or equal to the UDP payload length (i.e., the Length value in the UDP header minus the UDP header size). This allows the use of trailers after the IPv6 packet, which are defined in the following sections.

4.1. Trailers

Teredo packets can carry a variable number of type-length-value (TLV) encoded trailers, of the following format (intended to be similar to the use of IPv6 options defined in [RFC2460] section 4.2):



Type (1 byte): 8-bit identifier of the type of trailer.

Length (1 byte): 8-bit unsigned integer. Length of the Value field of this trailer, in octets.

Value (variable): Trailer-Type-specific data.

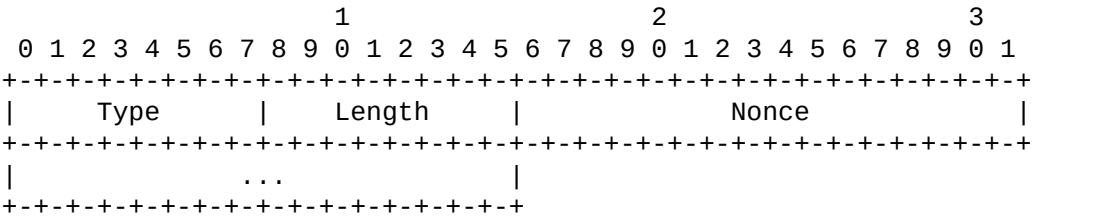
The trailer Type identifiers are internally encoded such that their highest-order two bits specify the action that is to be taken if the host does not recognize the trailer Type:

00, 10, 11 - skip over this trailer and continue processing the packet.

01 - discard the packet.

4.2. Nonce Trailer

The Nonce Trailer is used by the Symmetric NAT Support Extension (and therefore the UPnP-enabled Symmetric NAT Extension and Port-Preserving Symmetric NAT Extension also) and the Hairpinning Extension. The Nonce Trailer can be present in both indirect and direct bubbles. The nonce in the Nonce Trailer helps authenticate a Teredo client positioned behind a Symmetric NAT.



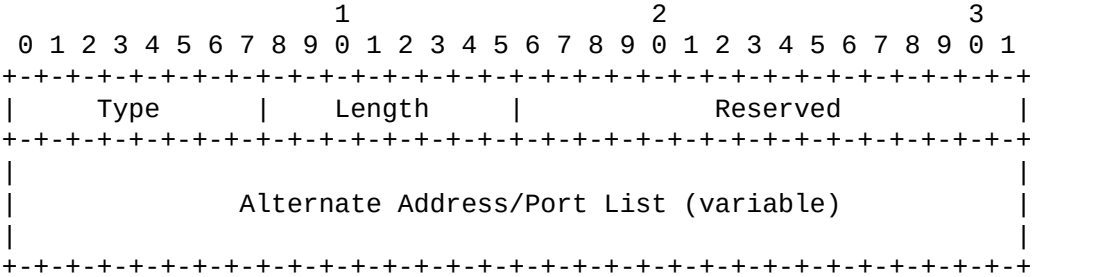
Type (1 byte): The Trailer Option type. This field MUST be set to 0x01.

Length (1 byte): The length in bytes of the rest of the option. This field MUST be set to 0x04.

Nonce (4 bytes): The nonce value.

4.3. Alternate Address Trailer

The Alternate Address Trailer is used by the Hairpinning Extension. The Alternate Address Trailer MUST NOT be present in any packets other than indirect bubbles sent by a Teredo client. The Alternate Address Trailer provides another Teredo client positioned behind the same NAT with more address options that it can use to connect.

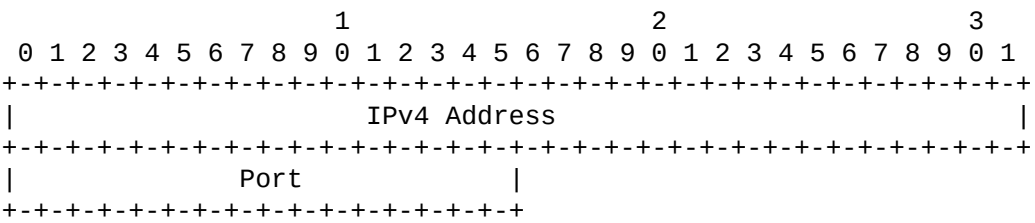


Type (1 byte): The Trailer Option type. This field MUST be set to 0x03.

Length (1 byte): The length in bytes of the rest of the option. The value of this field MUST be in the range 8 to 26 (i.e., 2 bytes for the Reserved field, and 6 bytes for each entry in the Alternate Address/Port List). This allows for a minimum of one address/port mapping and a maximum of four address/port mappings to be advertised. It SHOULD be at most 14 as a maximum of two address/port mappings can be determined by Teredo: one local address/port and one obtained using UPnP. Because the length of the alternate address/port is 6 bytes, the valid range of values is only 8, 14, 20, and 26.

Reserved (2 bytes): This field MUST be set to 0x0000 and ignored on receipt.

Alternate Address/Port List (variable): An array of additional address/port pairs that can be used by other Teredo clients to communicate with the sender. Each alternate address/port entry MUST be formatted as follows:

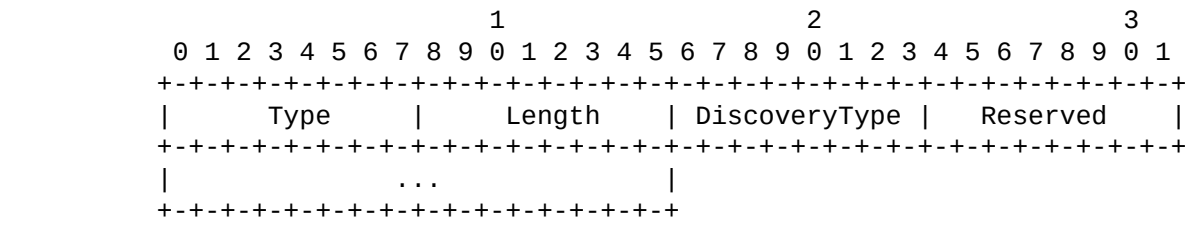


IPv4 Address (4 bytes): An IPv4 address in network byte order. This field MUST contain a valid unicast address.

Port (2 bytes): A port number in network byte order. This field MUST NOT be zero.

4.4. Neighbor Discovery Option Trailer

The Neighbor Discovery Option Trailer is used by the Server Load Reduction Extension because it allows direct bubbles to encode an IPv6 Neighbor Solicitation (Section 4.3 of [RFC4861]), in addition to an IPv6 Neighbor Advertisement (Section 4.4 of [RFC4861]). This allows packets to be sent without having to relay them through a Teredo server. The Neighbor Discovery Option Trailer allows the receiver to differentiate between a direct bubble that is soliciting a response versus a regular direct bubble. This allows Teredo clients to use direct bubbles to refresh inactive connections instead of using indirect bubbles.



Type (1 byte): The Trailer Option type. This field MUST be set to 0x04.

Length (1 byte): The length in bytes of the rest of the option. This field MUST be set to 0x04.

DiscoveryType (1 byte): This field MUST be set to one of the following values:

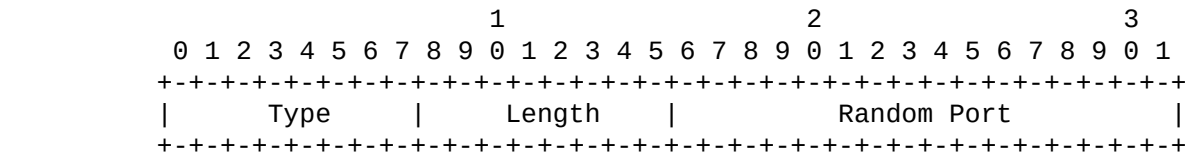
TeredoDiscoverySolicitation (0x00): The receiver is requested to respond with a direct bubble of DiscoveryType TeredoDiscoveryAdvertisement.

TeredoDiscoveryAdvertisement (0x01): The direct bubble is in response to a direct bubble or an indirect bubbles containing DiscoveryType TeredoDiscoverySolicitation.

Reserved (3 bytes): This field MUST be set to 0x000000 on transmission and ignored on receipt.

4.5. Random Port Trailer

The Random Port Trailer is used by the Port-Preserving Symmetric NAT Extension in both indirect and direct bubbles.



Type (1 byte): The Trailer Option type. This field MUST be set to 0x05.

Length (1 byte): The length in bytes of the rest of the option. This field MUST be set to 0x02.

Random Port (2 bytes): The external port that the sender predicts that its NAT has assigned it for communication with the destination. This field MUST be specified in network byte order.

5. Protocol Details

5.1. Common Processing

The behavior in this section applies to multiple extensions.

Packets equivalent to those sent for a peer the first time a connection is being established MAY be generated at other implementation-specific times. (For example, an implementation might choose to do so when its Neighbor Cache Entry for the peer is in the PROBE state.)

5.1.1. Refresh Interval

Section 5.2 of [RFC4380] states:

> 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.

This requirement can be problematic when the client is behind a NAT that expires state in less than 30 seconds. The optional interval determination procedure (Section 5.2.7 of [RFC4380]) also does not provide for intervals under 30 seconds. Hence, this document refines the behavior by saying the initial parameter SHOULD be configurable and the default MUST be 30 seconds. An implementation MAY set the randomized refresh interval to a value randomly chosen within an implementation-specific range. Such a range MUST fall within 50% to 150% of the refresh interval.

Section 5.2.5 of [RFC4380] states that:

> 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 send a router solicitation message to the server, as specified in Section 5.2.1;

This document refines the behavior as follows. A Teredo client MAY choose to send additional router solicitation messages to the server at other implementation-specific times. (For example, an implementation might choose to do so when its Neighbor Cache Entry for the router is in the PROBE state.)

5.1.2. Trailer Processing

A Teredo client MUST process the sequence of trailers in the same order as they appear in the packet. If the Teredo client does not recognize the trailer Type while processing the trailers in the Teredo packet, the client MUST discard the packet if the highest-order bits of the trailer Type contain 01, or else the Teredo client MUST skip past the trailer. A Teredo client MUST stop processing the trailers as soon as a malformed trailer appears in the sequence of trailers in the packet. A trailer is defined as malformed if it has any of the following properties:

- The length in bytes of the remainder of the UDP datagram is less than 2 (the size of the Type and Length fields of a trailer).

- The length in bytes of the remainder of the UDP datagram is less than 2 + the value of the Length field of the trailer.

5.2. Symmetric NAT Support Extension

Section 5.2.1 of [RFC4380] advises that no Teredo IPv6 address be configured if the Teredo client is positioned behind a symmetric NAT. For Teredo clients positioned behind symmetric NATs, the mapped address/port used by its NAT when communicating with a Teredo peer is different from the mapped address/port embedded in the Teredo client's Teredo IPv6 address. The Symmetric NAT Support Extension provides a solution to this problem.

In addition, Section 5.2.9 of [RFC4380] specifies a direct IPv6 connectivity test to determine that the mapped address/port in the Teredo IPv6 address of a peer is not spoofed. It does this through the use of a nonce in ICMPv6 Echo Request and Response messages (which are defined in Section 4 of [RFC4443]). However, the direct IPv6 connectivity test is limited only to communication between Teredo IPv6 addresses and non-Teredo IPv6 addresses. In the following extension, we introduce the use of a nonce in direct and indirect bubbles and provide a mechanism to verify that the mapped address/port are not spoofed.

This extension is optional; an implementation SHOULD support it.

5.2.1. Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

In addition to the state specified in Section 5.2 of [RFC4380], the following are also required.

Peer Entry: The following additional state is required on a per-peer basis:

- Nonce Sent: The value of the nonce sent in the last indirect bubble sent to the Teredo peer.

- Nonce Received: The value of the nonce received in the last indirect bubble received from the Teredo peer.

5.2.2. Timers

No timers are necessary other than those in [RFC4380].

5.2.3. Initialization

No initialization is necessary other than that specified in [RFC4380].

5.2.4. Message Processing

Except as specified in the following sections, the rules for message processing are as specified in [RFC4380].

5.2.4.1. Sending an Indirect Bubble

The rules for when indirect bubbles are sent to a Teredo peer are specified in Section 5.2.6 of [RFC4380]. When a Teredo client sends an indirect bubble, it MUST generate a random 4-byte value and include it in the Nonce field of a Nonce Trailer (Section 4.2) appended to the indirect bubble, and also store it in the Nonce Sent field of its Peer Entry for that Teredo peer.

5.2.4.2. Sending a Direct Bubble

The rules for when direct bubbles are sent to a Teredo peer are specified in Section 5.2.6 of [RFC4380]. When a Teredo client sends a direct bubble to a peer after receiving an indirect bubble with a Nonce Trailer, it MUST include in the direct bubble a Nonce Trailer with the same nonce value.

If the Teredo client is about to send a direct bubble before it has received an indirect bubble from the Teredo peer, the Teredo client MUST NOT include a Nonce Trailer.

5.2.4.3. Receiving an Indirect Bubble

The rules for processing an indirect bubble are specified in Section 5.2.3 of [RFC4380]. In addition, when a Teredo client receives an indirect bubble containing a Nonce Trailer, the Teredo client MUST store the nonce in the Nonce Received field of its Peer Entry for that Teredo peer. If an indirect bubble is received without a Nonce Trailer, and the Nonce Received field in the Peer Entry is non-zero, the Nonce Received field SHOULD be set to zero.

5.2.4.4. Receiving a Direct Bubble

If the mapped address/port of the direct bubble matches the mapped address/port embedded in the source Teredo IPv6 address, the direct bubble MUST be accepted, as specified in Section 5.2.3 of [RFC4380].

In addition, if the mapped address/port does not match the embedded address/port but the direct bubble contains a Nonce Trailer with a nonce that matches the Nonce Sent field of the Teredo peer, the direct bubble MUST be accepted.

If neither of the above conditions is true, the direct bubble MUST be dropped.

If the direct bubble is accepted, the Teredo client MUST record the mapped address/port from which the direct bubble is received in the mapped address/port fields of the Teredo peer, as specified in Section 5.2 of [RFC4380].

5.3. UPnP-Enabled Symmetric NAT Extension

The UPnP-enabled Symmetric NAT Extension is optional; an implementation SHOULD support it. This extension has the Symmetric NAT Support Extension (Section 5.2) as a dependency. Any node that implements this extension MUST also implement the Symmetric NAT Support Extension.

5.3.1. Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

This extension extends the abstract data model in Section 5.2.1 by adding the following additional fields.

UPnP-Enabled NAT flag: This is a Boolean value, set to TRUE if the NAT positioned in front of the Teredo client is UPnP enabled. The default value of this flag is FALSE.

UPnP-Mapped Address/Port: The mapped address/port assigned via UPnP to the Teredo client by the UPnP-enabled NAT behind which the Teredo client is positioned. Note that this field has a valid value only if the NAT to which the Teredo client is connected is UPnP enabled. Also, note that if the Teredo client is positioned behind a single NAT only (as opposed to a series of nested NATs), this value is the same as the mapped address/port embedded in its Teredo IPv6 address.

Symmetric NAT flag: This is a Boolean value, set to TRUE if the Teredo client is positioned behind a symmetric NAT.

Peer Entry: The following state needs to be added on a per-peer basis:

- Symmetric Peer flag: This is a Boolean value and is TRUE if the Teredo peer is positioned behind a symmetric NAT.

A Teredo client SHOULD also maintain the following state that is persisted across reboots:

- Persisted UPnP-Mapped Port: The mapped port assigned via UPnP to the Teredo client by the UPnP-enabled NAT behind which the Teredo client is positioned. Note that this value is the same as the UPnP-Mapped Port value when both are non-zero. The default value is all zero bytes.

5.3.2. Timers

No timers are necessary other than those in [RFC4380].

5.3.3. Initialization

Prior to beginning the qualification procedure, the Teredo client MUST first perform the uninitialization procedure specified in Section 5.3.5.1 if the Persisted UPnP-Mapped Port is supported and non-zero.

The Teredo client MUST then invoke the AddPortMapping function, as specified in Section 2.4.16 of [UPNPWANIP], with the following parameters:

- NewRemoteHost: "" (empty string)
- NewExternalPort: Local Port value
- NewProtocol: UDP
- NewInternalPort: Local Port value
- NewInternalClient: Local Address value
- NewEnabled: TRUE
- NewPortMappingDescription: "TEREDO"
- NewLeaseDuration: 0

The successful completion of the AddPortMapping function indicates that the NAT has created a port mapping from the external port of the NAT to the internal port of the Teredo client node. The parameters are specified so that any external host should be able to send packets to the Teredo client by sending packets to the mapped address/port. If the AddPortMapping function fails, the Teredo client MUST continue without using this extension. Otherwise, it MUST proceed as follows.

The Teredo client MUST set the UPnP-Mapped Port (and Persisted UPnP-Mapped Port, if supported) to the Local Port value specified in AddPortMapping. The Teredo client MUST then call the GetExternalIPAddress function specified in Section 2.4.18 of [UPNPWANIP]. If the GetExternalIPAddress function fails, the Teredo client SHOULD perform the uninitialization procedure specified in Section 5.3.5.1 and continue without using this extension. If the GetExternalIPAddress function succeeds, the Teredo client MUST proceed as follows.

The Teredo client MUST set the UPnP-Mapped Address to the address returned from the GetExternalIPAddress function, and set the UPnP-Enabled NAT flag to TRUE.

During the qualification procedure (as specified in Section 5.2.1 of [RFC4380]) when the Teredo client receives a response from the secondary Teredo server, the Teredo client MUST compare the mapped address/port learned from the secondary Teredo server with the mapped address/port associated with the Teredo server. If either the mapped address or the mapped port value is different, the Symmetric NAT flag MUST be set to TRUE.

After the qualification procedure, the mapped address/port learned from the Teredo server MUST be compared to the UPnP-Mapped Address/Port. If both are the same, the Teredo client is positioned behind a single NAT and the UPnP-Mapped Address/Port MUST be zeroed out.

5.3.4. Message Processing

Except as specified in the following sections, the rules for message processing are as specified in Section 5.2.3 of [RFC4380].

5.3.4.1. Receiving a Direct Bubble

Except as indicated below, the rules for handling a direct bubble are as specified in Section 5.2.4.4.

A Teredo client positioned behind a UPnP-enabled NAT (port-restricted NAT as well as symmetric NAT) will receive all packets sent to the mapped address/port embedded in its Teredo IPv6 address. Thus, when a Teredo client receives a direct bubble, it MUST compare the mapped address/port from which the packet was received with the mapped address/port embedded in the Teredo IPv6 address in the source address field of the IPv6 header. If the two are not the same, it indicates that the Teredo peer is positioned behind a symmetric NAT, and it MUST set the Symmetric Peer flag in its Peer Entry.

5.3.4.2. Sending a Direct Bubble

The rules for sending a direct bubble are specified in Section 5.2.6 of [RFC4380] and Section 5.2.4.2 of this document. These rules are further refined as follows.

If the Teredo client sending the direct bubble meets all of the following criteria:

- The Symmetric NAT flag is set to TRUE.

- The UPnP-Enabled NAT flag is set to TRUE.
- The UPnP-Mapped Address/Port are set to zero.
- The peer's Symmetric Peer flag is set to TRUE.

then the Teredo client MUST send the direct bubble to the mapped address/port embedded in the peer's Teredo IPv6 address.

This is because Symmetric-to-Symmetric and Port-Restricted-to-Symmetric NAT communication between the Teredo client and the peer would have failed anyway. However, by taking a chance that the peer might also be positioned behind a UPnP-enabled NAT just like the Teredo client itself, the Teredo client can try sending the direct bubble to the mapped address/port in the peer's Teredo IPv6 address. If the packet does go through, communication is established.

5.3.4.3. Sending a Data Packet

The rules for sending a data packet are specified in Section 5.2.4 of [RFC4380]. These rules are further refined as follows.

If the Teredo client sending the data packet meets all of the following criteria:

- The Symmetric NAT flag is set to TRUE.
- The UPnP-Enabled NAT flag is set to TRUE.
- The UPnP-Mapped Address/Port are set to zero.
- The peer's Symmetric Peer flag is set to TRUE.

then the Teredo client MUST send the data packet to the mapped address/port embedded in the peer's Teredo IPv6 address.

5.3.5. Shutdown

When Teredo client functionality is being shut down, uninitialization MUST be performed as specified in Section 5.3.5.1.

5.3.5.1. Uninitialization

First determine the mapped port as follows. If Persisted UPnP-Mapped Port is supported, use it as the mapped port. Otherwise, use the UPnP-Mapped Port.

If the mapped port is non-zero, the Teredo client MUST call the DeletePortMapping function, as specified in Section 2.4.17 of [UPNPWANIP], with the following parameters:

- NewRemoteHost: "" (empty string)
- NewExternalPort: the mapped port
- NewProtocol: UDP

5.4. Port-Preserving Symmetric NAT Extension

The Port-Preserving Symmetric NAT Extension is optional; an implementation SHOULD support it. This extension has the Symmetric NAT Support Extension (as specified in Section 5.2) as a dependency. Any node that implements this extension MUST also implement the Symmetric NAT Support Extension.

5.4.1. Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The Port-Preserving Symmetric NAT Extension extends the abstract data model in Section 5.2.1 by adding the following additional fields.

Port-Preserving NAT flag: This is a Boolean value, set to TRUE if the Teredo client is positioned behind a port-preserving NAT.

Symmetric NAT flag: This is a Boolean value, set to TRUE if the Teredo client is positioned behind a symmetric NAT.

Peer Entry: The following fields need to be added on a per-peer basis:

- Random Port: This field contains the value of the external port that the Teredo client predicts that its NAT has assigned it for communication with the peer. Set to zero by default.
- Peer Random Port: This field contains the value of the random port that the peer is using for communication with this Teredo client. Set to zero by default.

- Direct Receive on Primary Port: This is a Boolean value, set to TRUE if a packet is received from the Teredo peer on the primary local port. Set to FALSE by default.

- Direct Receive on Random Port: This is a Boolean value, set to TRUE if a packet is received from the Teredo peer on the Random Port. Set to FALSE by default.

- Connection Refresh Count: This field contains the number of direct bubbles that have been sent to the peer since the last time data was sent to the peer.

- Last Data Packet Sent Timestamp: This field contains the timestamp of the last data packet sent to the peer. This timestamp is different from the field that stores the data and time of last transmission to the peer (as specified in Section 5.2 of [RFC4380]) because the RFC-defined field is also updated every time a direct bubble is sent.

5.4.2. Timers

Other than those in [RFC4380], the Port-Preserving Symmetric NAT Extension requires the following additional timer.

Peer Refresh Timer: A timer to refresh peer connections through the random port, on which no data has been sent for a while.

5.4.2.1. Peer Refresh Timer Expiry

When the Peer Refresh Timer expires, the Teredo client MUST go through its list of peers and for each peer to which the Teredo client is communicating through the random port, the Teredo client MUST check the Last Data Packet Sent Timestamp to determine if data has been sent to the peer in the last 30 seconds, and check the Connection Refresh Count field to determine if the count has reached the maximum allowed value of 20. If both checks are FALSE, the Teredo client MUST send a direct bubble (as specified in Section 5.4.4.3) to the peer and increment the Connection Refresh Count. This direct bubble is sent as an attempt to keep the port mappings on all the intermediate NATs alive while the application/user may be temporarily inactive. If on the other hand, data has been sent to the peer in the last 30 seconds, the Connection Refresh Count MUST be reset to zero.

The Peer Refresh Timer MUST then be rescheduled to expire in 30 seconds.

5.4.3. Initialization

In addition to the behavior specified in [RFC4380], the Port-Preserving NAT flag and Symmetric NAT flag MUST be set to FALSE when the Teredo client is started. The Peer Refresh Timer MUST be started and scheduled to expire in 30 seconds.

During the qualification procedure (as specified in Section 5.2.1 of [RFC4380]), when the Teredo client receives a response from the Teredo server address, the Teredo client MUST compare the Port value in the origin indication, as specified in Section 5.1.1 of [RFC4380], with the Local Port value. If both values match, the client MUST set the Port-Preserving NAT flag to TRUE.

5.4.4. Message Processing

5.4.4.1. Sending a Data Packet

On receiving a data packet to be transmitted to the Teredo peer (in addition to the rules specified in Section 5.2.4 of [RFC4380]), the Teredo client MUST update the Last Data Packet Sent Timestamp when the packet is actually sent.

5.4.4.2. Sending an Indirect Bubble

The rules for sending an indirect bubble are as specified in Section 5.2.4.1 of this document and Section 5.2.6 of [RFC4380]. In addition to those rules, if the Port-Preserving NAT flag is TRUE, the Teredo client MUST do the following:

- If the Symmetric NAT flag is set, the Teredo peer is not marked as "trusted" (as specified in Section 5.2 of [RFC4380]), and the Random Port is zero, the Teredo client MUST first select a random port number to use, and then begin listening on that port. Since the NAT is port-preserving, the Teredo client can predict that the external port assigned will be equal to the random port chosen, and hence the Teredo client MUST store the random port chosen in the Random Port field of the Peer Entry.

- If the Random Port value is non-zero, the Teredo client MUST append a Random Port Trailer to the indirect bubble.

5.4.4.3. Sending a Direct Bubble

The rules for when direct bubbles are sent to a Teredo peer are as specified in Section 5.2.6 of [RFC4380]. In addition, Section 5.2.4.2 defines rules for enabling communication for clients positioned behind a symmetric NAT. In addition to the rules defined in both the aforementioned sections, if the Port-Preserving NAT flag is TRUE, the following rules apply also.

If the Symmetric NAT flag is set, and the Teredo peer is not marked as "trusted" (as specified in Section 5.2 of [RFC4380]) the Teredo client MUST send a direct bubble destined to the mapped address/port embedded in the Teredo IPv6 address of the Teredo peer. If the peer Random Port field is non-zero, the Teredo client MUST send another direct bubble from its own random port, destined to the peer random port. The IPv4 destination address MUST be the mapped address embedded in the Teredo IPv6 address. In addition, the Teredo client MUST include the Random Port Trailer (Section 4.5).

5.4.4.4. Receiving an Indirect Bubble

The rules for processing an indirect bubble are as specified in Section 5.2.4.3 of this document and Section 5.2.3 of [RFC4380]. In addition to these rules, if the incoming indirect bubble has a Random Port Trailer, the following additional processing MUST be done.

If the Peer Random Port field of the Peer Entry is zero, the Teredo client MUST store the port from the Random Port Trailer in the Peer Random Port field of the Peer Entry.

If the Peer Random Port field is non-zero and if either the Peer Random Port field and the new advertised port have the same value, or if active data has been exchanged between the two Teredo clients in the last 30 seconds (that is, "time of last transmission" or "time of last reception", as specified in Section 5.2 of [RFC4380], is set to a time that is less than 30 seconds ago), the new advertised port value MUST be ignored.

If the Peer Random Port field is non-zero and the new advertised port value is different from the Peer Random Port value, and it has been more than 30 seconds since the last exchange of data packets between the two Teredo clients, (that is, "time of last transmission" and "time of last reception" are set to a time that is more than 30 seconds ago), the Teredo client SHOULD store the new advertised port value in the Peer Random Port field and, if the Port-Preserving NAT flag is TRUE, then clear the Random Port field, and stop listening on the old random port. This allows communication to be re-established if either side changes the random port that it is using.

5.4.4.5. Receiving a Direct Bubble

The rules for handling direct bubbles are specified in Section 5.2.4.4 of this document and Section 5.2.3 of [RFC4380]. The rules for whether to accept a direct bubble are extended as follows, when the Port-Preserving NAT flag is TRUE:

- If the direct bubble is received on the primary port and the Teredo peer is not "trusted", the status field of the Teredo client MUST be changed to "trusted" and the Direct Receive on Primary Port flag MUST be set to TRUE. The mapped address/port from which the direct bubble was received MUST be recorded in the mapped address/port fields of the Teredo peer, as specified in Section 5.2 of [RFC4380]. The Teredo client MUST then set the Random Port field in the Peer Entry to zero and stop listening on the old random port.

- If the direct bubble is received on the primary port, the Teredo peer is "trusted", and the Direct Receive on Primary flag is set to TRUE, the Teredo client MUST compare the mapped address/port of the direct bubble with the mapped address/port of the Peer Entry. If both mappings are the same, the direct bubble MUST be accepted. If the mappings are different and it has been more than 30 seconds since the last packet exchange with the Teredo peer (that is, "time of last transmission" and "time of last reception", as defined in Section 5.2 of [RFC4380], are set to a time that is more than 30 seconds ago), the mapping on the Teredo peer's NAT has changed and communication needs to be re-established. This MUST be done by changing the status of the peer to "not-trusted", setting the Direct Receive on Primary Port flag to FALSE, and sending an indirect bubble to the Teredo peer via its Teredo server.

- If the direct bubble is received on the primary port, the Teredo peer is "trusted", the Direct Receive on Primary Port flag is set to FALSE, and the Direct Receive on Random Port flag is set to TRUE, the mapped address/port from which the direct bubble is received MUST be stored in the mapped address/port fields of the Peer Entry. The Direct Receive on Primary Port flag MUST be set to TRUE. The Teredo client MUST then set the Random Port field in the Peer Entry to zero and stop listening on the old random port. Finally, the Direct Receive on Random Port flag MUST be set to FALSE.

- If the direct bubble is received on the random port and the Teredo peer is not "trusted", the status field of the Teredo client MUST be changed to "trusted" and the Direct Receive on Random Port flag MUST be set to TRUE. The mapped address/port from which the direct bubble was received MUST be recorded in the mapped address/port fields of the Teredo Peer Entry, as specified in Section 5.2 of [RFC4380].

- If the direct bubble is received on the random port, the Teredo peer is "trusted", and the Direct Receive on Primary Port flag is FALSE, the Teredo client MUST compare the mapped address/port in the direct bubble with the mapped address/port in the Peer Entry. If the two mappings are the same, the direct bubble MUST be accepted. If the mappings are different, it implies that the NAT had deleted the mapping and when it reassigned the mapping, a different external port was chosen. In this instance, the Teredo client SHOULD set the Random Port field to zero, stop listening on the old random port, and send an indirect bubble to the Teredo peer as specified in Section 5.4.4.2.

Note that once the Direct Receive on Primary Port flag is TRUE, the client will stop listening on the random port and hence a direct bubble cannot be received on the random port. As a result, this case is intentionally omitted above.

5.5. Sequential Port-Symmetric NAT Extension

The Sequential Port-Symmetric NAT Extension is optional; an implementation SHOULD support it. This extension has the Symmetric NAT Support Extension (Section 5.2) as a dependency. Any node that implements this extension MUST also implement the Symmetric NAT Support Extension, as well as the Port-Preserving NAT Extension (Section 5.4).

5.5.1. Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The Sequential Port-Symmetric NAT Extension extends the abstract data model in Section 5.4.1 by adding the following additional state.

Peer Entry: The following fields need to be added on a per-peer basis:

- EchoTestNonce1: The value of the nonce sent as part of the authentication encapsulation, as specified in Section 5.1.1 of [RFC4380], in the router solicitation packet sent to the Teredo server address as part of the Echo Test.

- EchoTestNonce2: The value of the nonce sent as part of the authentication encapsulation in the router solicitation packet sent to the secondary Teredo server address as part of the Echo Test.

- EchoTestLowerPort: The value of the external port mapping extracted from the origin indication of the router advertisement received from the Teredo server address as part of the Echo Test. A value of 0 indicates that no such router advertisement has been received.

- EchoTestUpperPort: The value of the external port mapping extracted from the origin indication of the router advertisement received from the secondary Teredo server address as part of the Echo Test. A value of 0 indicates that no such router advertisement has been received.

- EchoTestRetryCounter: The number of times an Echo Test has been attempted.

5.5.2. Timers

In addition to the timers specified in Section 5.4.2, the following additional timer is required per Peer Entry.

Echo Test Failover Timer: A one-shot timer that runs whenever an Echo Test is in progress.

5.5.2.1. Peer Refresh Timer Expiry

The processing of the Peer Refresh Timer Expiry MUST be completed as specified in Section 5.4.2.1. In addition to those rules, the Teredo client MUST set the EchoTestLowerPort, EchoTestUpperPort, and EchoTestRetryCounter to zero.

5.5.2.2. Echo Test Failover Timer Expiry

If the Echo Test Failover Timer expires, the Teredo client MUST do the following.

If the value of the EchoTestRetryCounter is two, then the Teredo client MUST send an indirect bubble as specified in Section 5.2.4.1.

If the value of the EchoTestRetryCounter is one, then the Teredo client MUST start another Echo Test as specified in Section 5.5.4.1.1.

5.5.3. Initialization

No behavior changes are required beyond what is specified in Section 5.4.3.

5.5.4. Message Processing

Except as specified in the following sections, the rules for message processing are as specified in Section 5.4.4.

5.5.4.1. Handling a Request to Send an Indirect Bubble

Whenever [RFC4380] or other extensions specified in this document specify that an indirect bubble is to be sent, the following actions apply at that time instead if the Symmetric NAT flag is TRUE and the Port-Preserving NAT flag is FALSE. Note that any behavior specified by [RFC4380] or other extensions in this document still applies to how indirect bubbles are constructed, but such behavior is done at a later time as specified in Section 5.5.4.4.

If the Symmetric NAT flag is TRUE, and the Port-Preserving NAT flag is FALSE, and the Teredo peer is not marked as "trusted" (as specified in Section 5.2 of [RFC4380]), and the Random Port is zero, then the Teredo client MUST select a random port number to use, begin listening on that port, and start an Echo Test as specified below.

5.5.4.1.1. Starting an Echo Test

To start an Echo Test, the Teredo client MUST send the following three packets from this port:

- First, a router solicitation (as specified in Section 5.2.1 of [RFC4380]) MUST be sent to the Teredo server address. The router solicitation MUST include an authentication encapsulation with a randomly generated Nonce field, as specified in Section 5.1.1 of [RFC4380]. The nonce included in the authentication encapsulation MUST then be stored in the EchoTestNonce1 field of the Peer Entry.

- Second, a direct bubble MUST be sent to the peer.

- Third, a router solicitation MUST be sent to the secondary Teredo server address. The router solicitation MUST include an authentication encapsulation with a randomly generated Nonce field, as specified in Section 5.1.1 of [RFC4380]. The nonce included in the authentication encapsulation MUST then be stored in the EchoTestNonce2 field of the Peer Entry.

The Teredo client MUST then increment the EchoTestRetryCounter and set the Echo Test Failover Timer to expire in a number of seconds equal to EchoTestRetryCounter.

5.5.4.2. Sending an Indirect Bubble

The rules for sending an indirect bubble are as specified in Section 5.2.4.1 of this document and Section 5.2.6 of [RFC4380]. In addition to those rules, if the Symmetric NAT flag is TRUE, and the Port-Preserving NAT flag is FALSE, and the Random Port value is non-zero, then the Teredo client MUST append a Random Port Trailer to the indirect bubble.

5.5.4.3. Receiving a Direct Bubble

The processing of the direct bubble MUST be completed as specified in Section 5.4.4.5, as if the Port-Preserving NAT flag were TRUE. After the processing is complete, if the Direct Bubble Received on Primary flag is TRUE, and the Echo Test Failover Timer is running, then the Echo Test Failover Timer MUST be canceled and EchoTestLowerPort, EchoTestUpperPort, and EchoTestRetryCounter MUST be set to zero.

5.5.4.4. Receiving a Router Advertisement

The rules for processing a router advertisement are as specified in Section 5.2.1 of [RFC4380]. In addition to those rules, if the router advertisement contains an authentication encapsulation, the Teredo client MUST look for a Peer Entry whose EchoTestNonce1 or EchoTestNonce2 field matches the nonce in the authentication encapsulation. If a Peer Entry is found, the Teredo client MUST do the following.

If the received nonce is equal to EchoTestNonce1 and EchoTestLowerPort is zero, then EchoTestLowerPort MUST be set to the external port mapping extracted from the origin indication of this router advertisement.

If the received nonce is equal to EchoTestNonce2 and EchoTestUpperPort is zero, then EchoTestUpperPort MUST be set to the external port mapping extracted from the origin indication of this router advertisement.

If the EchoTestUpperPort and EchoTestLowerPort are now both non-zero, the Teredo client MUST then set the Random Port field of the Peer Entry to (EchoTestUpperPort + EchoTestUpperPort)/2, rounded down, and send an indirect bubble as specified in Section 5.5.4.2.

5.6. Hairpinning Extension

This extension is optional; an implementation SHOULD support it.

5.6.1. Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

In addition to the state specified in Section 5.2 of [RFC4380], the following are also required:

UPnP Mapped Address/Port: The mapped address/port assigned via UPnP to the Teredo client by the UPnP-enabled NAT behind which the Teredo client is positioned. This field has a valid value only if the NAT to which the Teredo client is connected is UPnP enabled. In addition, if the Teredo client is positioned behind a single NAT only (as opposed to a series of nested NATs), this value will be the same as the mapped address/port embedded in its Teredo IPv6 address.

Peer Entry: Per-peer state is extended beyond what is described in [RFC4380] by including the following:

- Alternate Address/Port list: The list of alternate address/port pairs advertised by the peer.

5.6.2. Timers

No timers are necessary other than those in [RFC4380].

5.6.3. Initialization

Behavior is as specified in [RFC4380], with the following additions.

Prior to beginning the qualification procedure, the Teredo client MUST invoke the AddPortMapping function (as specified in Section 2.4.16 of [UPNPWANIP]) with the parameters specified in Section 5.3.3. If successful, it indicates that the NAT has created a port mapping from the external port of the NAT to the internal port

of the Teredo client node. If the AddPortMapping function is successful, the Teredo client MUST store the mapping assigned by the NAT in its UPnP Mapped Address/Port state.

After the qualification procedure, the mapped address/port learned from the Teredo server MUST be compared to the UPnP Mapped Address/Port. If both are the same, the Teredo client is positioned behind a single NAT and the UPnP Mapped Address/Port MUST be zeroed out.

5.6.4. Message Processing

5.6.4.1. Sending an Indirect Bubble

The rules for when indirect bubbles are sent to a Teredo peer are as specified in Section 5.2.6 of [RFC4380]. If communication between a Teredo client and a Teredo peer has not been established, the Teredo client MUST include the Alternate Address Trailer in the indirect bubble. The Alternate Address Trailer MUST include the node's local address/port in the Alternate Address/Port list. If the UPnP Mapped Address/Port is non-zero, the Alternate Address Trailer MUST also include it in the list.

Hairpinning requires "direct IPv6 connectivity tests" (as specified in Section 5.2.9 of [RFC4380]) to succeed before it can accept packets from an IPv4 address and port not embedded in the Teredo IPv6 address. Hence, the indirect bubble MUST also include a Nonce Trailer.

5.6.4.2. Receiving an Indirect Bubble

The rules for processing indirect bubbles are as specified in Section 5.2.3 of [RFC4380]. In addition to those rules, when a Teredo client receives an indirect bubble with the Alternate Address Trailer, it SHOULD first verify that the Alternate Address Trailer is correctly formed (as specified in Section 4.3), and drop the bubble if not. Otherwise, it MUST set the Alternate Address/Port list in its Peer Entry to the list in the trailer. The Teredo client, besides sending direct bubbles to the mapped address/port embedded in the Teredo IPv6 address (as specified in Section 5.2.6 of [RFC4380]), MUST also send a direct bubble to each mapped address/port advertised in the Alternate Address Trailer.

In each of the direct bubbles, the Teredo client MUST include a Nonce Trailer with the nonce value received in the indirect bubble.

5.6.4.3. Receiving a Direct Bubble

If the mapped address/port of the direct bubble matches the mapped address/port embedded in the source Teredo IPv6 address, the direct bubble MUST be accepted, as specified in Section 5.2.3 of [RFC4380].

If the mapped address/port does not match the embedded address/port, but the direct bubble contains a Nonce Trailer with a nonce that matches the Nonce Sent field of the Teredo peer, the direct bubble MUST be accepted.

If neither of the above rules match, the direct bubble MUST be dropped.

5.7. Server Load Reduction Extension

This extension is optional; an implementation SHOULD support it.

5.7.1. Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

In addition to the state specified in Section 5.2 of [RFC4380], the following are also required.

Peer Entry: The following state needs to be added on a per-peer basis:

- Count of Solicitations Transmitted: The number of Solicitation packets sent.

5.7.2. Timers

Retransmission Timer: A timer used to retransmit Teredo Neighbor Solicitation packets.

When the retransmission timer expires, the Teredo client MUST retransmit a direct bubble with a Neighbor Discovery Option Trailer, and increment the Count of Solicitations Transmitted. If the count is less than three, it MUST then reset the timer to expire in two seconds. Otherwise (if the count is now three), it MUST send an

indirect bubble to the Teredo peer to re-establish connectivity as if no communication between the Teredo client and the Teredo peer had been established.

5.7.3. Initialization

No initialization is necessary other than that specified in [RFC4380].

5.7.4. Message Processing

Except as specified below, processing is the same as specified in [RFC4380].

5.7.4.1. Sending a Data Packet

Upon receiving a data packet to be transmitted to the Teredo peer, the Teredo client MUST determine whether data has been exchanged between the Teredo client and peer in either direction in the last 30 seconds (using the state as specified in Section 5.2 of [RFC4380]). If not, the Teredo client MUST send a direct bubble with a Neighbor Discovery Option Trailer having the DiscoveryType field set to TeredoDiscoverySolicitation. The Count of Solicitations Transmitted field MUST be set to 1. The retransmission timer MUST be set to expire in two seconds.

5.7.4.2. Receiving a Direct Bubble

The rules for processing direct bubbles are as specified in Section 5.2.3 of [RFC4380]. In addition to those rules, upon receiving a direct bubble containing a Neighbor Discovery Option Trailer with DiscoveryType field set to TeredoDiscoverySolicitation, the Teredo client MUST respond with a direct bubble with the Neighbor Discovery Option Trailer having the DiscoveryType field set to TeredoDiscoveryAdvertisement.

6. Protocol Examples

The following sections describe several operations as used in common scenarios to illustrate the function of Teredo Extensions.

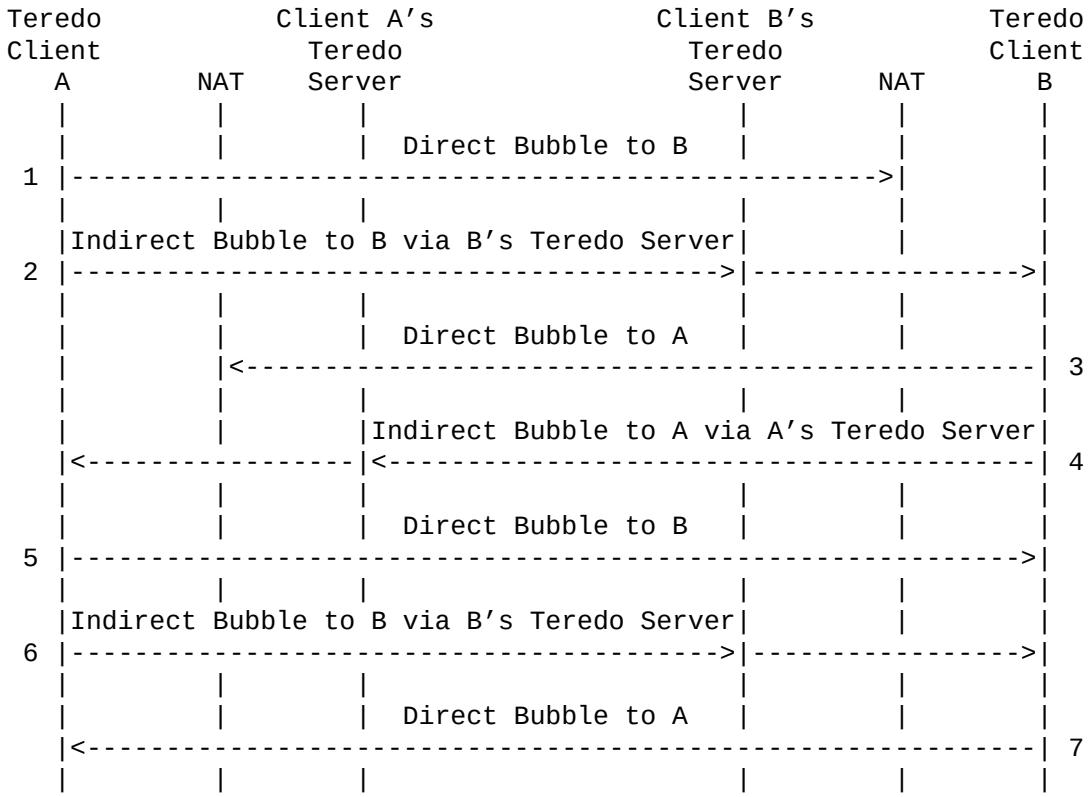
6.1. Symmetric NAT Support Extension

The following protocol example illustrates the use of the Symmetric NAT Support Extension.

In Figure 2 (Section 3.1), assume that Teredo Client A, which is positioned behind a port-symmetric NAT, wants to communicate with Teredo Client B, which is positioned behind an address-restricted NAT.

The qualification procedure where the Teredo client determines that it is positioned behind a symmetric NAT is exactly the same as that specified in Section 5.2.1 of [RFC4380]. Because of the Symmetric NAT Extension, Client A continues to configure a Teredo IPv6 address even after determining that the Teredo client is positioned behind a symmetric NAT.

Next the following packet exchange helps Teredo Client A (A) establish communication with Teredo Client B (B).



Port-Symmetric NAT to Address-Restricted NAT Packet Exchange

1. A sends a direct bubble (Packet 1) destined to the mapped address/port embedded in B's Teredo IPv6 address. The mapped port in the source field of the packet assigned by Client A's NAT is different from the mapped port embedded in A's Teredo IPv6 address. This is characteristic of the port-symmetric NAT positioned in front of A. The mapped address in the source field of the packet is the same as the mapped address embedded in the Teredo IPv6 address of A.

2. The aforementioned direct bubble is dropped by B's NAT because it has not seen an outgoing packet destined to A's mapped IPv4 address.

3. A sends an indirect bubble (Packet 2) destined to B via Client B's Teredo server.

4. The above-mentioned indirect bubble is received by B. B then responds with the following packets. The first packet sent by B is a direct bubble (Packet 3) destined to the mapped address/ port embedded in A's Teredo IPv6 address.

5. The above-mentioned direct bubble is dropped by A's NAT because the NAT has not seen any outgoing packet sourced from the mapped address/port embedded in A's Teredo IPv6 address and destined to the mapped address/port embedded in B's Teredo IPv6 address.

6. B also sends an indirect bubble (Packet 4) destined to A via A's Teredo Server.

7. The aforementioned indirect bubble is successfully received by A. A responds to the indirect bubble with its own direct bubble (Packet 5). This direct bubble is exactly the same as the first direct bubble (Packet 1) sent by A.

8. This time around the aforementioned direct bubble is accepted by B's NAT because the NAT has seen an outgoing packet (Packet 3) sourced from the mapped address/port embedded in B's Teredo IPv6 address and destined to the mapped address/port embedded in A's Teredo IPv6 address. It is important to remember that A's NAT is port-symmetric and therefore varies only the mapped port while the mapped address remains the same. B's NAT is address-restricted and cares only about prior communication with the IPv4 address, not the specific port. At this point, communication in one direction is now possible (B to A, but not vice versa).

9. After receiving the direct bubble, B remembers the new mapped address/port that was in the source fields of the direct bubble and uses those for future communication with A instead of the mapped address/port embedded in A's Teredo IPv6 address.

10. A then times out and resends an indirect bubble (Packet 6) and in response, B sends a direct bubble (Packet 7). This direct bubble is destined to the new learned mapped address/port and hence A's NAT permits the direct bubble through. Communication is now possible in the other direction (client A to B).

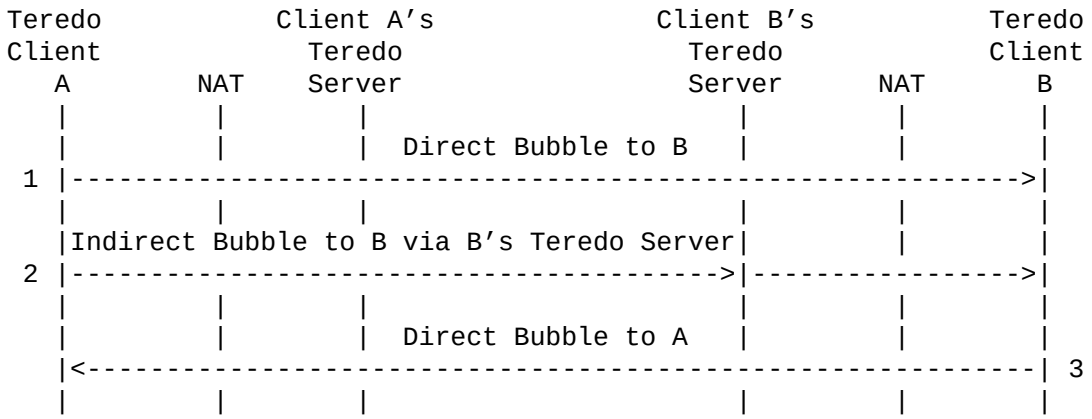
6.2. UPnP-Enabled Symmetric NAT Extension

The following protocol example illustrates the use of the UPnP-Enabled Symmetric NAT Extension in addition to the Symmetric NAT Support Extension.

Assume that Teredo Client A, which is positioned behind a UPnP-enabled port-symmetric NAT, wants to communicate with Teredo Client B, which is also positioned behind a UPnP-Enabled port-symmetric NAT.

Before both clients start their qualification procedure, they use UPnP to reserve port mappings on their respective NATs. The UPnP operations succeed for both the clients and the clients hence know that they are positioned behind UPnP-enabled NATs. After the qualification procedure, both clients have valid Teredo IPv6 addresses because they both support the Symmetric NAT Support Extension. Also, after the qualification procedure both clients will compare their mapped address/port determined through UPnP with the mapped address/port determined through the qualification procedure. Because both will be the same, the clients will zero out their UPnP mapped address/port values and conclude that they are each located behind a single UPnP-enabled NAT.

The following packet exchange shows Teredo client A (A) establishing communication with Teredo client B (B).



UPnP-enabled Symmetric NAT Packet Exchange

1. A sends a direct bubble (Packet 1) to the mapped address/port embedded in B's Teredo IPv6 address. Because A's NAT is a symmetric NAT, the UDP source port field in the packet assigned by A's NAT is different from the mapped port embedded in A's Teredo IPv6 address, but the IPv4 source address of the packet is the same as the mapped address embedded in A's Teredo IPv6 address.

2. The above-mentioned direct bubble is received by B because it is destined for the UPnP mapped address/port of B and hence is let through by the NAT. At this point, B deduces that A is positioned behind a symmetric NAT because the mapped address/port from which the direct bubble is received is different from the mapped address/port that is embedded in A's Teredo IPv6 address. Hence, it remembers that the peer is positioned behind a symmetric NAT so that data packets will be sent to the mapped address/port embedded in A's Teredo IPv6 address, rather than the mapped address/port from which the direct bubble was received. At this point, communication in one direction is now possible (B to A, but not vice versa).

3. A also sends an indirect bubble (Packet 2) destined to B via B's Teredo Server.

4. The above indirect bubble is received by B. B then responds with a direct bubble (Packet 3) destined to the mapped address/port embedded in A's Teredo IPv6 address, as in step 2.

5. Because A's NAT is also UPnP enabled, the above-mentioned direct bubble is received by A. A also notices that B is positioned behind a Symmetric NAT because the mapped address/port from which the packet is received is different from the mapped address/port

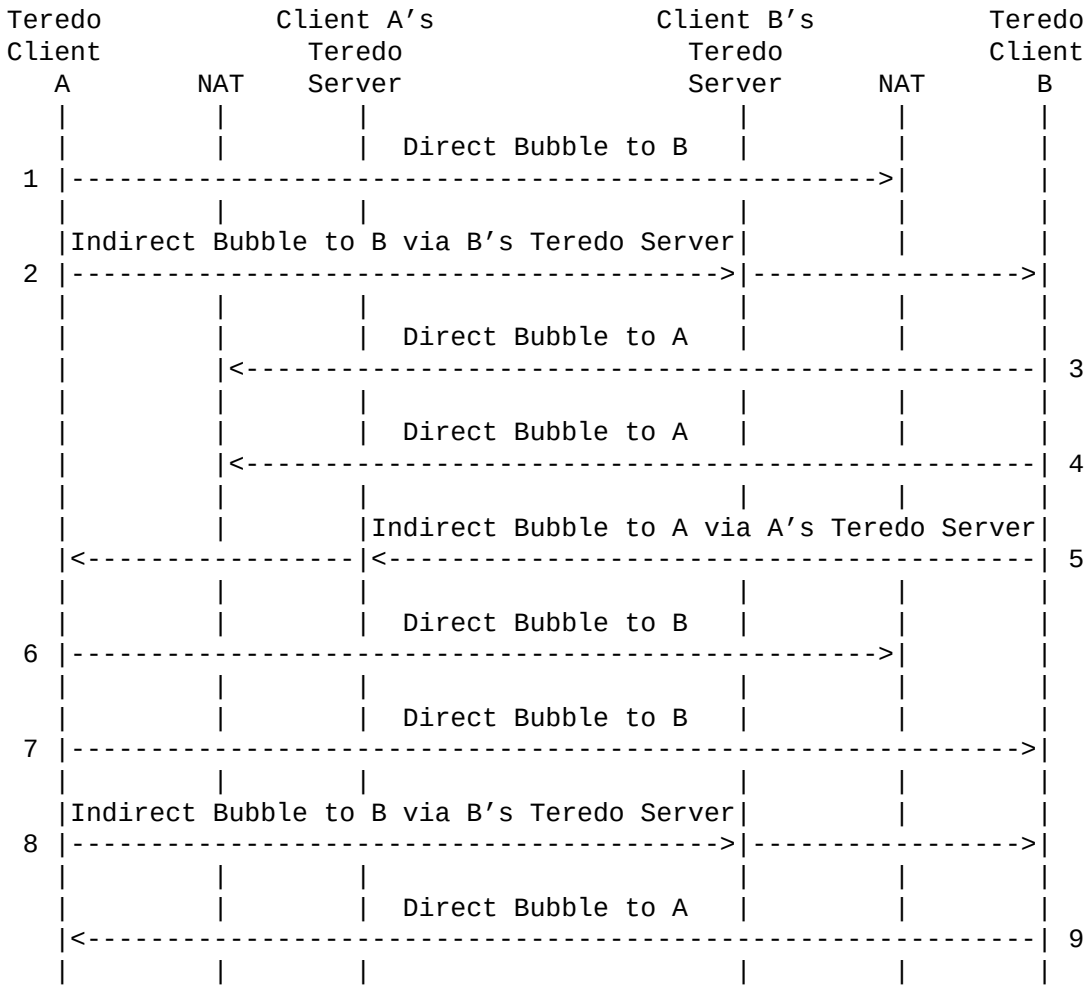
embedded in B's Teredo IPv6 address. Hence, it remembers that the peer is positioned behind a symmetric NAT so that data packets will be sent to the mapped address/port embedded in B's Teredo IPv6 address, rather than the mapped address/port from which the direct bubble was received. At this point, communication is now possible in the other direction (A to B).

6.3. Port-Preserving Symmetric NAT Extension

The following protocol example illustrates the use of the Port-Preserving Symmetric NAT Extension.

Assume that Teredo Client A (A), which is positioned behind a port-preserving symmetric NAT, wants to communicate with Teredo Client B (B), which is also positioned behind a port-preserving symmetric NAT.

The following packet exchange explains the configuration setup and communication setup between the two clients.



Port-Preserving Symmetric NAT Packet Exchange

1. During the qualification procedure, when the clients receive a response from the Teredo server, they compare the Port value in the Origin indication with the Local Port value. If both values match, the clients set the Port-Preserving NAT flag to TRUE.

2. When the response is received from the secondary Teredo server, the mapped address/port value in the Origin indication is compared with the mapped address/port value learned from the response received from the primary server. If the mappings are different, the Symmetric NAT flag is set to TRUE.

3. It is assumed that for both Clients A and B, the Port-Preserving NAT flag and the Symmetric NAT flag are set to TRUE at the end of the qualification procedure.

4. Before A sends packets to B, A checks to see if it is positioned behind a port-preserving NAT and a symmetric NAT, which in the example, it is. A also checks to see if the peer is "trusted", but it currently is not. Next, A checks if the Random Port is set to non-zero. Since it is still zero, A allocates a new random port, begins listening on it, and stores the value in the Random Port field.

5. A sends a direct bubble (Packet 1) from the primary port to the mapped address/port embedded in B's Teredo IPv6 address. This direct bubble does not have a Nonce Trailer or a Random Port Trailer attached to the end.

6. The aforementioned direct bubble is dropped by B's NAT because the NAT has not seen an outgoing packet destined to A's mapped address.

7. A sends an indirect bubble (Packet 2) destined to B via client B's Teredo server. This indirect bubble contains two trailers: the Nonce Trailer containing a random nonce, and the Random Port Trailer containing the random port value from the Peer Entry. The nonce used in the Nonce Trailer is also stored in the Nonce Sent field of the Peer Entry.

8. The aforementioned indirect bubble is received by B. B adds the Teredo peer to its peer list. B saves the nonce value from the Nonce Trailer in the Nonce Advertised field of the Peer Entry. B stores the port value from the Random Port Trailer in the Peer Random Port field in the Peer Entry.

9. B responds by sending the following packets. The first packet sent by B is a direct bubble (Packet 3) destined to the mapped address/port embedded in A's Teredo IPv6 address. This packet is sent from the primary port. It includes the Nonce Trailer with the nonce from the Nonce Advertised field of the Peer Entry.

10. The aforementioned direct bubble is dropped by A's NAT because the NAT has not seen any outgoing packet sourced from the mapped address/port embedded in A's Teredo IPv6 address and destined to the mapped address/port embedded in B's Teredo IPv6 address.

11. B then checks if it is positioned behind a port-restricted NAT or a symmetric NAT. It also checks if the peer has already advertised a random port. In this case, B is positioned behind a port-preserving symmetric NAT and the peer has advertised a random port; hence, it needs to use a random port. It checks if its Random Port field is set to non-zero. Since it is still

zero, B allocates a new random port, begins listening on it, and stores it in the Random Port entry of the Peer Entry. B then sends a direct bubble (Packet 4) destined to the mapped address embedded in A's Teredo IPv6 address and the port stored in the Peer Random Port field of the Peer Entry. The direct bubble is sent from its own random port.

12. The above direct bubble is dropped by A's NAT because the NAT has not seen any outgoing packet sourced from the mapped address embedded in A's Teredo IPv6 address and random port advertised by A.

13. B also sends an indirect bubble (Packet 5) destined to A via A's Teredo Server. This indirect bubble includes a Nonce Trailer and a Random Port Trailer. The Nonce Trailer includes a new randomly generated nonce that is also stored in the Nonce Sent field of the Peer Entry. The Random Port Trailer includes the value in the Random Port field of the Peer Entry.

14. The aforementioned indirect bubble is successfully received by A. A parses the trailers and stores the nonce contained in the Nonce Trailer in the Nonce Received field of the Peer Entry. A stores the port advertised in the Random Port Trailer in the Random Port field of the Peer Entry.

15. A responds with the following packets in response to the indirect bubble received. The first packet is a direct bubble (Packet 6) sent from the primary port and is destined to the mapped address/port embedded in B's Teredo IPv6 address.

16. The aforementioned direct bubble again is dropped by B's NAT because the NAT has not seen an outgoing packet with the same 4-tuple as the incoming packet.

17. The next packet is also a direct bubble (Packet 7) and this one is sent from A's random port. The packet is destined to the mapped address embedded in B's Teredo IPv6 address and the Peer Random Port stored in the Peer Entry.

18. Because both NATs are port-preserving NATs and the random ports have not been used for any other mapping, the aforementioned direct bubble is received by B because B's NAT has seen an outgoing packet (Packet 4) with the same address/port pairs. B stores the address/port from which the direct bubble was received in the mapped address/port fields of the Peer Entry. It changes the status of the peer to "trusted" and sets the

Direct Receive on Random Port field to TRUE. At this point, communication in one direction is now possible (B to A, but not vice versa).

19. Because A still considers B to be "not-trusted", it times out and retransmits an indirect bubble (Packet 8). This packet contains a new nonce as part of the Nonce Trailer and also contains the value of the random port as part of the Random Port Trailer.

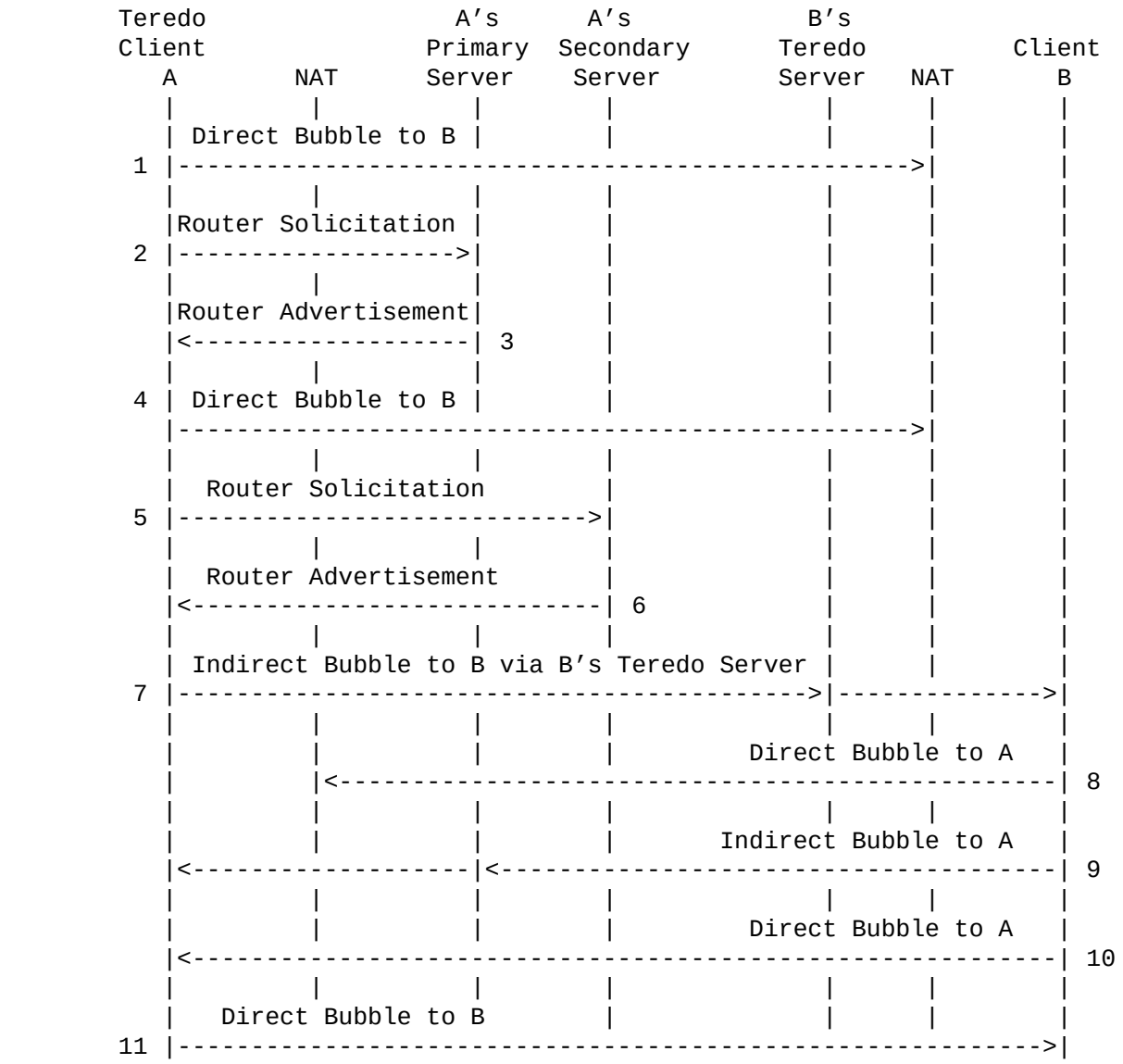
20. B receives the aforementioned indirect bubble. The processing of this indirect bubble is similar to the processing of Packet 2. Since B received a direct bubble on its random port, it does not respond with a direct bubble from its primary port. Instead, it responds with a direct bubble (Packet 9) sent from its random port, which is similar to Packet 4 mentioned above.

21. A receives the direct bubble sent by B. A stores the mapped address/port from which the direct bubble was received in mapped address/port fields in the Peer Entry. A changes the status of B to "trusted" and sets the Direct Receive on Random Port field to TRUE. At this point, the communication is now possible in the other direction (A to B).

6.4. Sequential Port-Symmetric NAT Extension

The following protocol example illustrates the use of the Sequential Port-Symmetric NAT Extension.

Assume that Teredo Client A (A), which is positioned behind a sequential port-symmetric NAT and implements the Sequential Port-Symmetric NAT Extension, wants to communicate with Teredo Client B (B), which is positioned behind a port-restricted NAT that supports the Port-Preserving Port-Symmetric NAT Extension. The following packet exchange explains the configuration setup and communication setup between the two clients.



Sequential Port-Symmetric NAT Packet Exchange

1. During the qualification procedure, when Client A receives a response from the Teredo Server, it compares the Port value in the Origin indication with the Local Port value. Since they are different, it concludes that it is not behind a port-preserving NAT, and so assumes it is behind a sequential port-symmetric NAT.

2. When A wants to communicate with B, A starts by sending a direct bubble (Packet 1) from its primary port. This occurs because Client A does not know Client B's NAT type, which could be a cone

or address restricted NAT or UPnP-enabled NAT. Because Client A is behind a symmetric NAT, the external port used by A's NAT is a new port. This direct bubble will be dropped by B's NAT since Client B is behind a port-restricted NAT.

3. Because Client A does not know if B is behind a port restricted NAT or some other kind of NAT, Client A proactively opens a new random internal port, say, port 1100.

4. Client A then performs its Echo Test as follows:

   A. Client A sends a router solicitation (Packet 2) to its Teredo Server address from port 1100. The server responds with a router advertisement (Packet 3).

   B. Client A sends a direct bubble (Packet 4) to the peer from port 1100 destined to the port advertised in Client B's Teredo address, say, port 2100. This direct bubble is dropped by Client B's port-restricted NAT.

   C. Client A sends a router solicitation (Packet 5) to its secondary Teredo server address from port 1100. The server responds with a router advertisement (Packet 6).

   D. On receiving the corresponding router advertisements for Packet 2 and Packet 4, Client A knows that port 1100 maps to, say, port 1200 for Packet 2 and port 1202 for Packet 4.

   E. Client A then calculates its predicted port used for Packet 2 as the average (rounded down) of 1200 and 1202, i.e., 1201.

5. Client A then sends out an indirect bubble (Packet 7). This indirect bubble contains a random port trailer that contains the predicted port, port 1201. This indirect bubble makes it to Client B.

6. Client B sends out the following bubbles in response to the indirect bubble:

   A. The first direct bubble (Packet 8) is destined for the port mapping embedded in Client A's Teredo Address. (It has been observed that some NATs display symmetric NAT behavior for outgoing packets but cone NAT behavior for incoming packets. The direct bubble described is likely to succeed if Client A's NAT displays such a behavior.) Since in this example, A's NAT is a normal sequential port-symmetric NAT, this packet is dropped.

B. The second packet is an indirect bubble (Packet 9) sent to Client A without any trailers since Client B is behind a port-restricted NAT.

C. The next packet will be a direct bubble (Packet 10) sent to port 1201. This packet will make it in to Client A since Client A previously sent an outgoing packet (Packet 4) with the same four tuple. At this point, communication in one direction is now possible (A to B, but not vice versa).

7. Client A then sends a direct bubble (Packet 11) to Client B when it receives Packet 10. This time, the bubble makes it through to B because it previously sent an outgoing packet (Packet 10) with the same four tuple. At this point, communication is now possible in the other direction (B to A).

6.5. Hairpinning Extension

The following protocol example illustrates the use of the Hairpinning Extension.

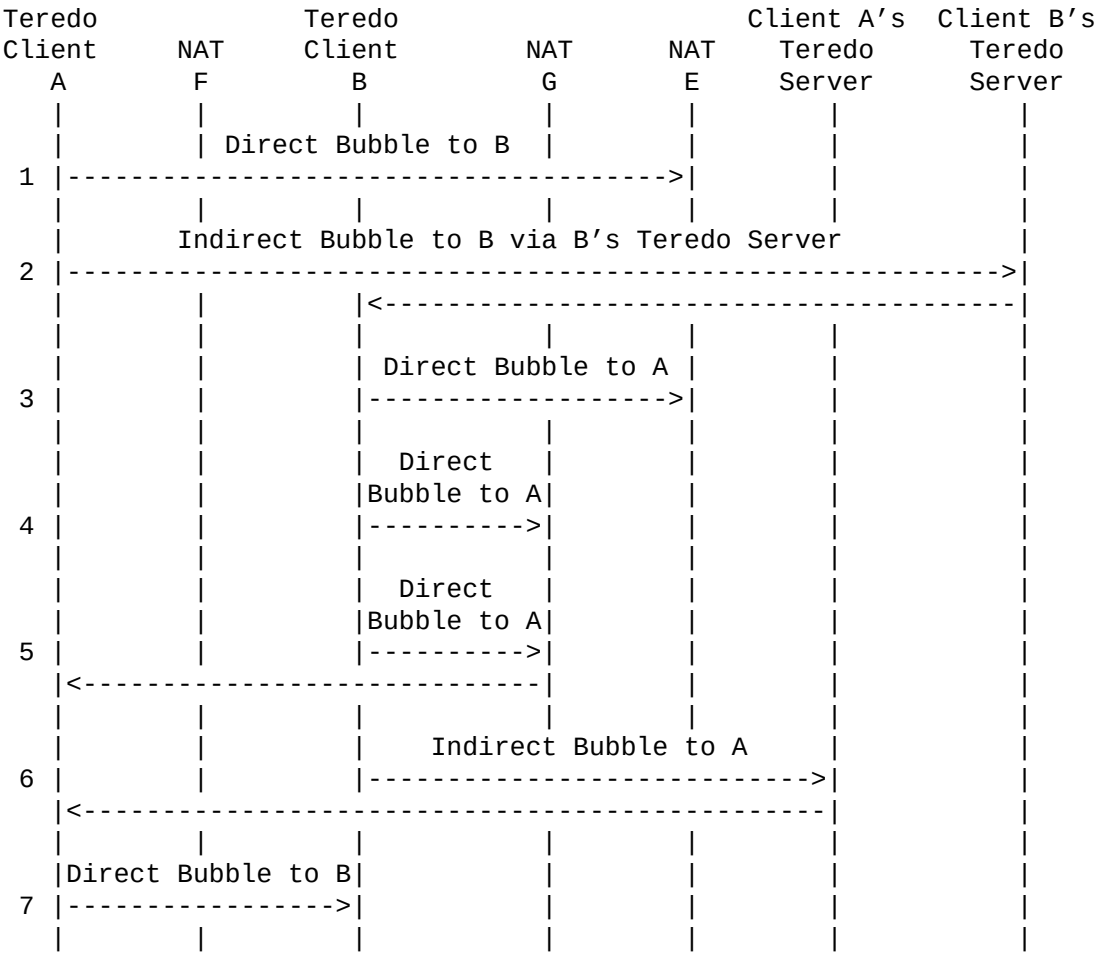
In Figure 3 (Section 3.5), Teredo Client A (A) and Teredo Client B (B) are positioned behind different immediate NATs in a two-layer NAT topology; that is, the outermost NAT (NAT E) is common to both A and B but the immediate NATs that they are connected to are different (A is connected to NAT F while B is connected to NAT G). Further assume that the immediate NATs that A and B are connected to are UPnP-enabled (NAT F and NAT G are UPnP-enabled). We assume that NAT E does not support hairpinning; that is, the NAT does not relay packets originating from the private address space and destined for the public address of the NAT, back to the private address of the NAT.

Before starting the qualification procedure, both A and B use UPnP to reserve port mappings on their respective NATs. They observe that the UPnP operation succeeds and both clients obtain valid UPnP Mapped Address/Port values.

Next, both client A and client B implement the qualification procedure where they determine their mapped address/port values, as specified in Section 5.2.1 of [RFC4380].

A and B both compare their UPnP Mapped Address/Port values with the mapped address/port values obtained through the qualification procedure. Because both A and B are part of a two-layer NAT topology, these values will be different. Hence, both A and B continue to hold on to their UPnP Mapped Address/Port.

The following packet exchange shows client A establishing communication with client B.



Hairpinning-Based Packet Exchange

1. A sends a direct bubble (Packet 1) to the mapped address/port embedded in B's Teredo IPv6 address.

2. The aforementioned direct bubble is dropped by NAT E, because it does not support Hairpinning.

3. A sends out an indirect bubble (Packet 2) destined to B via B's Teredo Server. In this indirect bubble, A includes an Alternate Address Trailer that includes both the local address/port and the UPnP mapped address/port.

4. The aforementioned indirect bubble is received by B. After parsing the Alternate Address Trailer, B has a total of three addresses to communicate with: two from the Alternate Address Trailer and one from the mapped address/port embedded in A's Teredo IPv6 address. B then responds with the following packets. The first packet sent by B is a direct bubble (Packet 3) destined to the mapped address/port embedded in A's Teredo IPv6 address.

5. The aforementioned direct bubble will be dropped by the NAT E because it does not support Hairpinning.

6. Because the local address/port was the first mapping in the Alternate Address Trailer, the second direct bubble (Packet 4) sent by B is destined to the local address/port.

7. The aforementioned direct bubble is dropped because A and B are positioned behind different NATs and hence have their own private address space. A's local address is not reachable from B.

8. The next direct bubble (Packet 5) is sent by B destined to A's UPnP mapped address/port, which is the second mapping in the Alternate Address Trailer sent by A.

9. The aforementioned direct bubble is received by A because A's UPnP-mapped address is reachable from B. A stores the source address from which the direct bubble was received in the mapped address/port fields of the Peer Entry, as defined in Section 5.2 of [RFC4380]. Also, the mapped address status field (as specified in Section 5.2.3 of [RFC4380]) is changed to "trusted". At this point, communication in one direction (A to B) is now possible, but not vice versa because B has not yet marked A as trusted.

10. B also sends an indirect bubble (Packet 6) to A via A's Teredo server. As part of the indirect bubble, B also includes an Alternate Address Trailer, which contains the local address/port and the UPnP mapped address/port of B.

11. The aforementioned indirect bubble is received by A. After parsing the Alternate Address Trailer, A adds the two addresses in the Alternate Address Trailer to the Alternate Address List in the Peer Entry. Because the peer's mapping is "trusted" (point 9), A responds with only one direct bubble (Packet 7) that is sent to the mapped address/port stored in the Peer Entry.

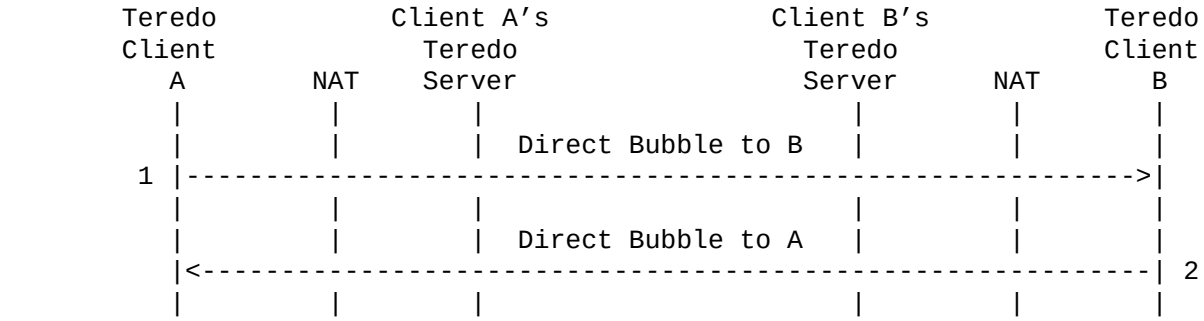
12. The aforementioned direct bubble is received by B. B records the mapped address/port from which the direct bubble was received in the mapped address/port field in its Peer Entry, and changes the status of the mapped address to "trusted". At this point, communication is now possible in the other direction (B to A).

6.6. Server Load Reduction Extension

The following protocol example illustrates the use of the Server Load Reduction Extension.

Assume that Teredo Client A (A) has established communication with Teredo Client B (B). Also, assume that at some later point when no data packets have been exchanged between both clients for more than 30 seconds, the communication needs to be re-established because A wants to send a data packet to B.

The following packet exchange helps A re-establish communication with B.



Server Load Reduction Packet Exchange

1. A sends a direct bubble (Packet 1) with the Neighbor Discovery Option Trailer, with the DiscoveryType field set to TeredoDiscoverySolicitation.

2. If the mapping on either of the NATs has not expired, the direct bubble is received by B. B parses the Neighbor Discovery Option and because the DiscoveryType was set to TeredoDiscoverySolicitation, B responds with a direct bubble (Packet 2). B's direct bubble also contains the Neighbor Discovery Option and the DiscoveryType is set to TeredoDiscoveryAdvertisement.

3. The aforementioned direct bubble is received by A and at this point, communication between the Teredo clients is re-established.

7. Security Considerations

Security considerations are the same as those specified in Section 7 of [RFC4380].

In addition, the Hairpinning Extension introduces the possibility of an amplification attack if a malicious user could advertise a large number of port mappings in the Alternate Address Trailer, resulting in a large number of direct bubbles sent in response. Because of this, Section 4.3 explicitly limits the number of addresses that a Teredo client will accept.

Because the nonce in the Nonce Trailer is used (as specified in Section 5.2.4.4) to prevent spoofing of bubbles that would result in directing traffic to the wrong place, it is important that the nonce be random so that attackers cannot predict its value. See [RFC4086] for further discussion of randomness requirements.

8. Acknowledgements

Thanks to Gurpreet Virdi and Poorna Gaddehosur for technical contributions to this document, and to the V6OPS WG and Jari Arkko for their helpful reviews.

9. IANA Considerations

IANA has created a new trailer Type registry. Requests for new trailer Type values are made through Specification Required [RFC5226]. Initial values are listed below.

Trailer Type	Usage	Reference
0x01	Nonce Trailer	RFC 6081
0x02	Random Port Trailer	RFC 6081
0x03	Alternate Address Trailer	RFC 6081
0x04	Neighbor Discovery Option Trailer	RFC 6081

10. References

10.1. Normative References

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

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

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[UPNPWANIP] UPnP Forum, "WANIPConnection:1", November 2001, <http://www.upnp.org/standardizeddcps/documents/UPnP_IGD_WANIPConnection%201.0.pdf>.

10.2. Informative References

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

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

Author's Address

Dave Thaler
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
USA

Phone: +1 425 703 8835
EMail: dthaler@microsoft.com