[MS-TERE]:
Teredo Extensions

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
04/03/2007	0.1		MCPP Milestone Longhorn Initial Availability
06/01/2007	0.1.1	Editorial	Revised and edited the technical content.
07/03/2007	1.0	Major	MLonghorn+90
07/20/2007	1.1	Minor	Updated the technical content.
08/10/2007	1.2	Minor	Updated the technical content.
09/28/2007	1.2.1	Editorial	Revised and edited the technical content.
10/23/2007	1.2.2	Editorial	Revised and edited the technical content.
11/30/2007	1.2.3	Editorial	Revised and edited the technical content.
01/25/2008	1.2.4	Editorial	Revised and edited the technical content.
03/14/2008	1.3	Minor	Updated the technical content.
05/16/2008	1.3.1	Editorial	Revised and edited the technical content.
06/20/2008	1.3.2	Editorial	Revised and edited the technical content.
07/25/2008	1.3.3	Editorial	Revised and edited the technical content.
08/29/2008	1.3.4	Editorial	Revised and edited the technical content.
10/24/2008	1.4	Minor	Updated the technical content.
12/05/2008	2.0	Major	Updated and revised the technical content.
01/16/2009	3.0	Major	Updated and revised the technical content.
02/27/2009	4.0	Major	Updated and revised the technical content.
04/10/2009	5.0	Major	Updated and revised the technical content.
05/22/2009	6.0	Major	Updated and revised the technical content.
07/02/2009	7.0	Major	Updated and revised the technical content.
08/14/2009	7.0.1	Editorial	Revised and edited the technical content.
09/25/2009	7.0.2	Editorial	Revised and edited the technical content.
11/06/2009	8.0	Major	Updated and revised the technical content.
12/18/2009	8.0.1	Editorial	Revised and edited the technical content.
01/29/2010	8.1	Minor	Updated the technical content.

Date	Revision History	Revision Class	Comments
03/12/2010	9.0	Major	Updated and revised the technical content.
04/23/2010	9.0.1	Editorial	Revised and edited the technical content.
06/04/2010	10.0	Major	Updated and revised the technical content.
07/16/2010	10.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	10.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	10.0	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	10.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

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

1.1 Glossary

The following terms are specific to this document:

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.

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.

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 [RFC4380] section 2.8. The **IPv4 Destination Address** and User Datagram Protocol (UDP) Destination Port fields contain a **mapped address/port**.

echo test: A mechanism to predict the **mapped address/port** that 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 packet behavior, 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 [RFC4380] section 5.2.4.

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 that is specified in [RFC4380]. For **symmetric NATs**, the mapped address/port can be different for every **peer** that a node tries to communicate with.

Network Address Translator (NAT): A device that converts between IP addresses used within an intranet (or other private network) and Internet IP addresses.

nonce: A time-variant counter 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.

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 [RFC1918] section 3.

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 that is 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.

Teredo bubble: A Teredo control message (specified in [RFC4380] section 2.8) 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.

Teredo IPv6 address: An IPv6 address that starts with the prefix 2001:0000:/32 and is formed as specified in [RFC4380] section 2.14.

Teredo secondary server address: A secondary IPv4 address of a **Teredo server** with which a **Teredo client** is configured, as specified in [RFC4380] section 5.2.

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 defined in [RFC4380] section 5.2.

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.)

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980, http://www.ietf.org/rfc/rfc768.txt

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., et al., "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996, http://www.ietf.org/rfc/rfc1918.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC2460] Deering, S., and Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998, http://www.ietf.org/rfc/rfc2460.txt

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006, http://www.ietf.org/rfc/rfc4380.txt

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and Soliman, H., "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007, http://www.ietf.org/rfc/rfc4861.txt

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

1.2.2 Informative References

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, November 1987, http://www.ietf.org/rfc/rfc1035.txt

[RFC2463] Conta, A., and Deering, S., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", December 1998, http://www.ietf.org/rfc/rfc2463.txt

[RFC3022] Srisuresh, P., and Egevang, K., "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001, http://www.ietf.org/rfc/rfc3022.txt

1.3 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 User Datagram Protocol (UDP).

When a node behind a NAT needs to communicate with a **peer** (that is, 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 (that is, 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**, are 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 aforementioned restriction.

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

Source NAT \ Destination NAT	Cone	Address restricted	Port restricted	UPnP Port restricted	UPnP Port symmetric	Port-preserving Port-symmetric	Sequential Port-symmetric	Port symmetric	Address-symmetric
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

Figure 1: Matrix of connectivity improvements for Teredo extensions

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.

1.3.1 Symmetric NAT Support Extension

The qualification procedure (as specified in [RFC4380] section 5.2.1) 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, [RFC4380] section 5.2.1 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 client 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 the following figure. Teredo Client B uses Teredo Server 2 to learn that its mapped address/port is 157.54.0.10:8192 and constructs a Teredo IPv6 address, as specified in [RFC4380] section 4. Hence, CE49:7601 is the hexadecimal value of the address of Teredo Server 2 (206.73.118.1), the mapped port is exclusive-OR'ed with 0xFFFF to form DFFF, and the Mapped Address is exclusive-OR'ed with 0xFFFFFFFF to form 62C9:FFF5.

Teredo Client A uses Teredo Server 1 to learn that its mapped address/port is 157.60.0.1:4096 and, with this extension, constructs a Teredo IPv6 address (as specified in [RFC4380] section 4), even though it learns that it is behind a symmetric NAT. Hence, CFD1:4478 is the hexadecimal value of the address of Teredo Server 1 (207.209.68.120), the mapped port is exclusive-OR'ed with 0xFFFF to form EFFF, and the Mapped Address is exclusive-OR'ed with 0xFFFFFFFF to form 62C3:FFFE.

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.

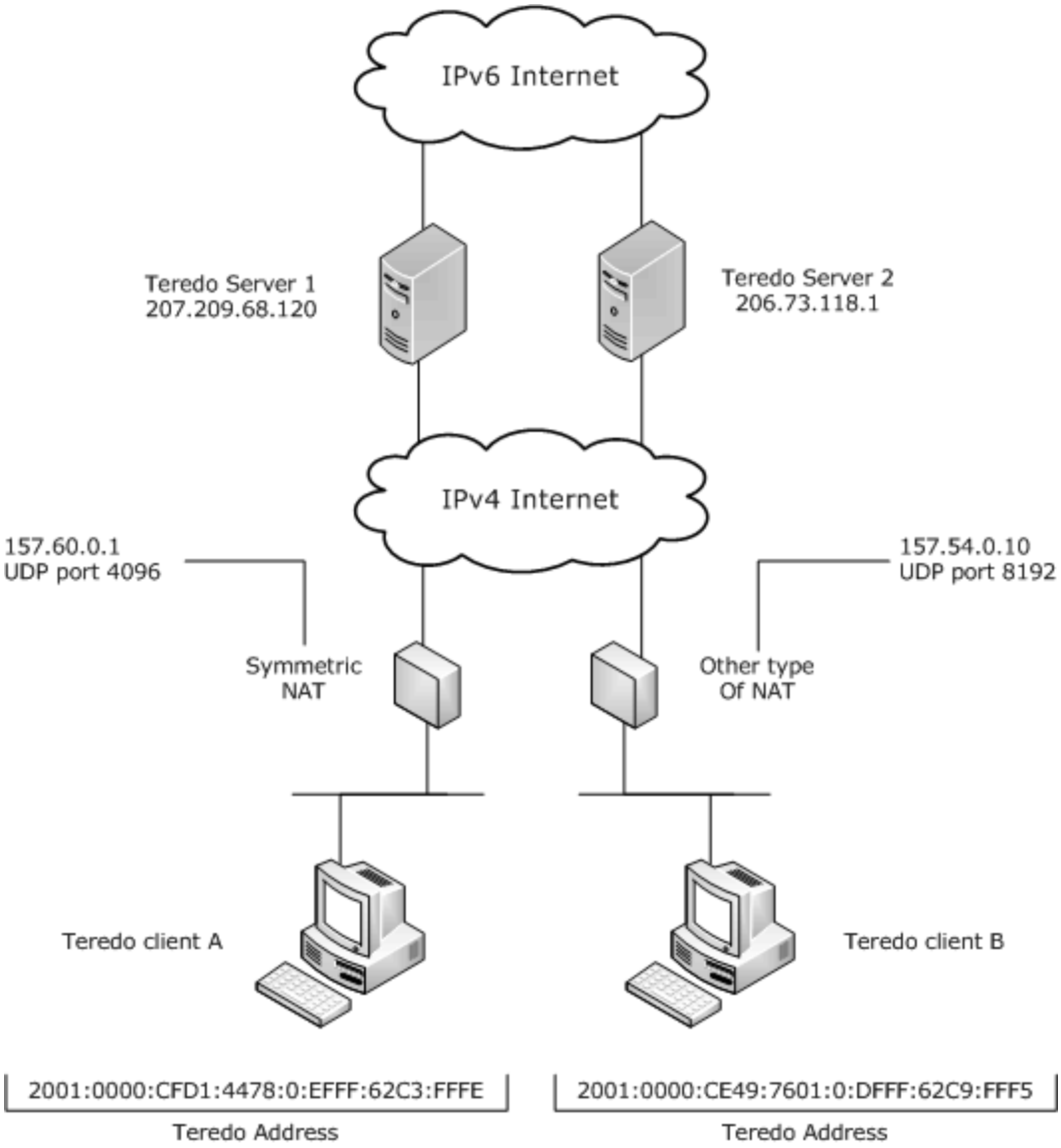


Figure 2: Symmetric NAT example

In the first case, assume that 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. Therefore, B 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.

1.3.2 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 the figure in section 1.3.1, 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 1.3.1, Client B also sends an indirect bubble that elicits a direct bubble from Client A. Unlike the case in section 1.3.1, 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 1.3.1, Client A's NAT was a cone or address-restricted NAT that 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 local address/port to mapped address/port translation. 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. However, it does not typically cause packets sourced from the local address/port to be translated to have the mapped address/port as the external source; hence, it 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, which is embedded in the Teredo IPv6 address. Outgoing packets instead will appear to come from a different mapped address/port due to the symmetric NAT behavior.

1.3.3 Port-Preserving Symmetric NAT Extension

The Port-Preserving Symmetric NAT Extension is dependent on the Symmetric NAT Support Extension (section 1.3.1). 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 1.3.2. Note that the Port-Preserving Symmetric NAT Extension described here is independent of the UPnP-enabled Symmetric NAT Extension, described in section 1.3.2.

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 1.3.1.

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.

1.3.4 Sequential Port-Symmetric NAT Extension

The Sequential Port-Symmetric NAT Extension is dependent on the Symmetric NAT Support Extension in section 1.3.1. 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**. However, the following assumptions apply.

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

- No foolproof method exists for discovering whether the client is behind a sequential port-symmetric NAT.
- These kinds of NATs are notorious for changing their behavior. Sometimes the NAT might be sequential port-symmetric and other times not.
- There is no other solution for symmetric NAT traversal, so this is a last resort.

Teredo clients that are positioned behind sequential port-symmetric NATs can also listen on a randomly chosen local port when they communicate with a peer. To predict the external port that a specific peer is using, the client sends three packets:

- Packet 1 is a router solicitation (as specified in [RFC4380] section 5.2.1) that is sent to the Teredo server address.
- Packet 2 is a direct bubble that is sent to the peer.
- Packet 3 is a router solicitation that is sent to the secondary Teredo server address.

As part of the usual Teredo service, the Teredo server responds to packets 1 and 3. Based on the information in the Teredo server 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. If 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. However, because other applications might also have used 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 actually the external port for packet 2, the direct bubble reaches the Teredo client.

1.3.5 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 [RFC4380] section 8.3.

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.

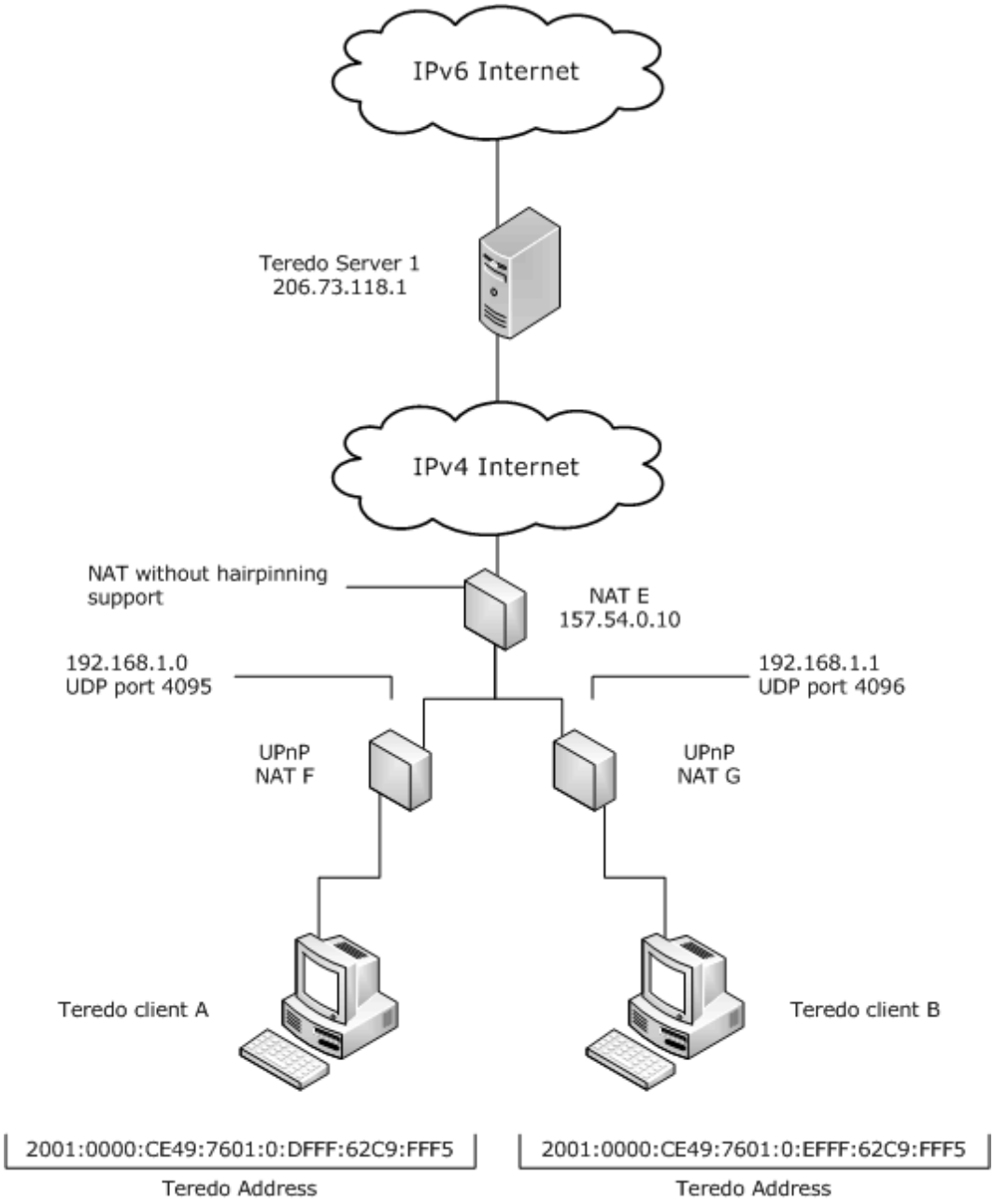


Figure 3: Hairpinning example

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.

1.3.6 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 [RFC4380] section 5.2.4. 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 reestablish 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 [RFC4380] section 5.2.4.

1.3.7 Random Address Extension

The **flags** field in a Teredo IPv6 address has 13 unused bits out of a total of 16 bits. To guard against address-scanning risks from malicious users, the Random Address Extension uses 12 of the 13 unused bits when configuring the Teredo IPv6 address. Even if a malicious user were able to determine the external (mapped) IPv4 address and port assigned to the Teredo client, the malicious user would still need to attack a range of 4,096 IPv6 addresses to determine the actual Teredo IPv6 address of the client.

1.4 Relationship to Other Protocols

The extensions specified in this document do not introduce any new protocol relationships beyond the Teredo protocol (as specified in [RFC4380]) except that the UPnP-enabled Symmetric NATs Extension and the Hairpinning Extension both make use of UPnP, as specified in [UPNPWANIP].

Teredo, as specified in [RFC4380], is used for providing IPv6 connectivity over an IPv4 Internet. Teredo relies on UDP, as specified in [RFC768], as a transport. Teredo relies on the capability to resolve names to addresses (for example, Domain Name System (DNS), as specified in [RFC1035]), although it does not depend on the existence of any particular name resolution protocol.

1.5 Prerequisites/Preconditions

These extensions have no prerequisites or preconditions except those that are specified in [RFC4380].

Use of these extensions assumes that the Teredo client has an IPv4 address and UDP connectivity, and that IPv6 functionality is enabled. The Teredo server address is also known<1> to the Teredo client at startup, which allows the Teredo client to determine its Teredo client address. Finally, it is

assumed that a Teredo client knows (or learns via some external mechanism, such as name resolution) the Teredo client address of the other Teredo client with which it wants to communicate.

1.6 Applicability Statement

The Symmetric NAT Support Extension (section 1.3.1) applies when a Teredo client is positioned behind a symmetric NAT.

The UPnP-enabled Symmetric NAT Extension (section 1.3.2) applies when a Teredo client positioned behind a UPnP-enabled symmetric NAT tries to communicate with another Teredo client that is positioned behind either a UPnP-enabled symmetric NAT or a UPnP-enabled port-restricted NAT.

The Port-Preserving Symmetric NAT Extension (section 1.3.3) applies when a Teredo client positioned behind a port-preserving symmetric NAT tries to communicate with another Teredo client that is positioned behind either a port-restricted NAT or a port-preserving symmetric NAT.

The Hairpinning Extension (section 1.3.5) applies when two Teredo clients are positioned behind the same NAT, and that NAT does not have hairpinning support.

The Server Load Reduction Extension (section 1.3.6) applies when communication between two Teredo clients goes silent for more than 30 seconds (that is, no data packets are exchanged for more than 30 seconds) and a Teredo client would send an indirect bubble (as specified in [RFC4380] section 5.2.6) as a result. Server load reduction helps avoid routing through the Teredo server to refresh the connection state on the two Teredo clients.

The Random Address Extension (section 1.3.7) is applicable in all Teredo scenarios.

1.7 Versioning and Capability Negotiation

The extensions in this document inherit the extensibility and capability negotiation features of Teredo, as specified in [RFC4380].

The Random Address Extension does not need to be negotiated (or detected) because each client can use it independently.

The Symmetric NAT Support Extension, the Hairpinning Extension, the Server Load Reduction Extension, the UPnP-enabled Symmetric NAT Extension, and the Port-Preserving Symmetric NAT Extension are active only when both ends of Teredo communication support the extension. Capability detection is specified under each extension section (see section 3).

1.8 Vendor-Extensible Fields

The Teredo extensions in this document do not introduce any new vendor-extensible fields.

1.9 Standards Assignments

There are no standards assignments other than what is specified in [RFC4380].

2 Messages

2.1 Transport

All Teredo messages are transported over UDP, as specified in [RFC4380] section 3.

2.2 Message Syntax

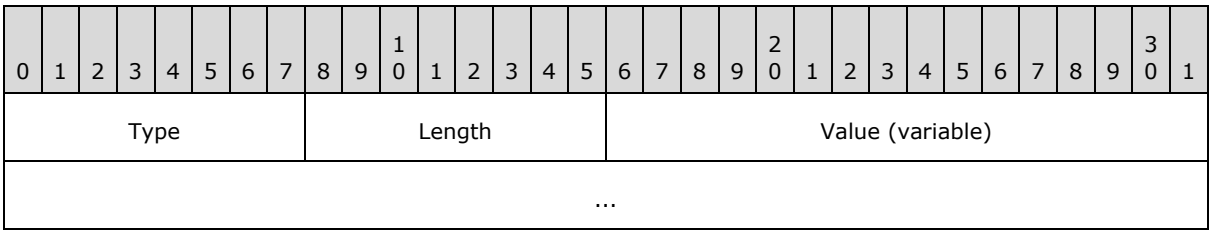
[RFC4380] section 5.2.3 states:

"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 to its own."

This document clarifies the word "consistent" as used in the previous statement as follows. The IPv6 payload length is "consistent" with the length of the UDP datagram if the IPv6 packet length (that is, the Payload Length value in the IPv6 header plus the IPv6 header size) is less than or equal to the UDP payload length (that is, 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.

2.2.1 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.

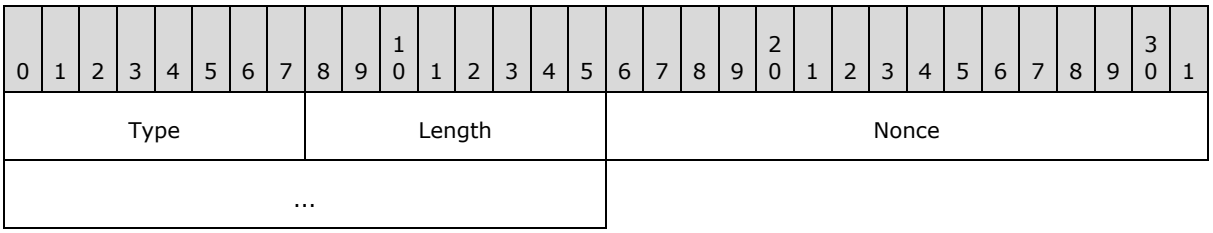
Value (variable): Trailer **Type**-specific data.

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.

2.2.1.1 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.



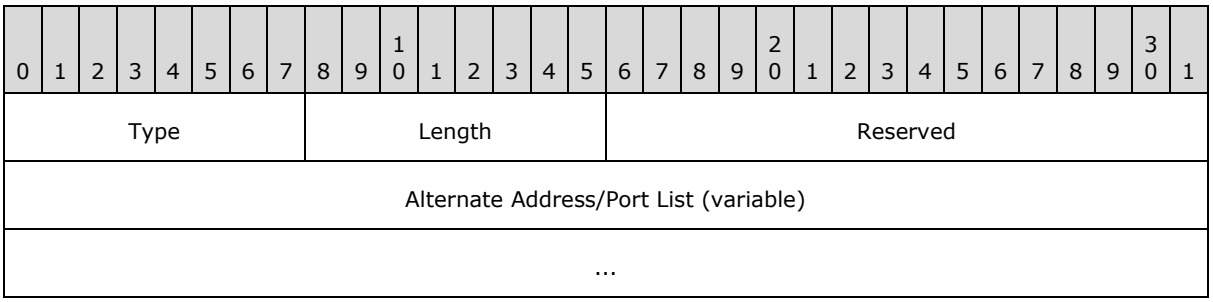
Type (1 byte): The Trailer Option type. This field MUST be set to 0x01.

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.

2.2.1.2 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.

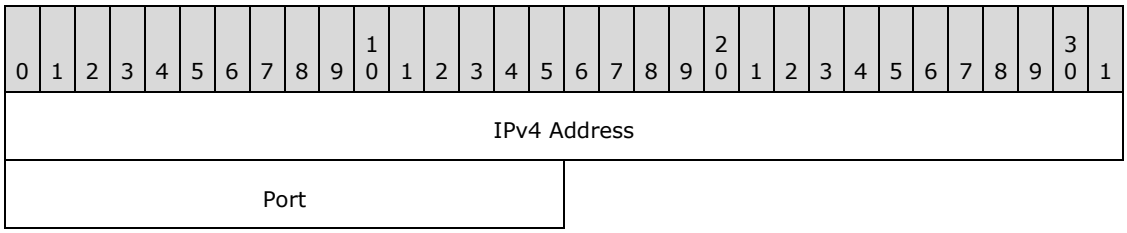
Length (1 byte): The length in bytes of the rest of the option. The value of this field SHOULD be in the range 8 to 26 (that is, 2 bytes for the **Reserved** field, and 6 bytes for each entry in the Alternate Address/Port List). This allows for a minimum of one address/port mapping and a maximum of four address/port mappings to be advertised. It SHOULD be at most 14, because 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): MUST be set to zero when sent and MUST be 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.

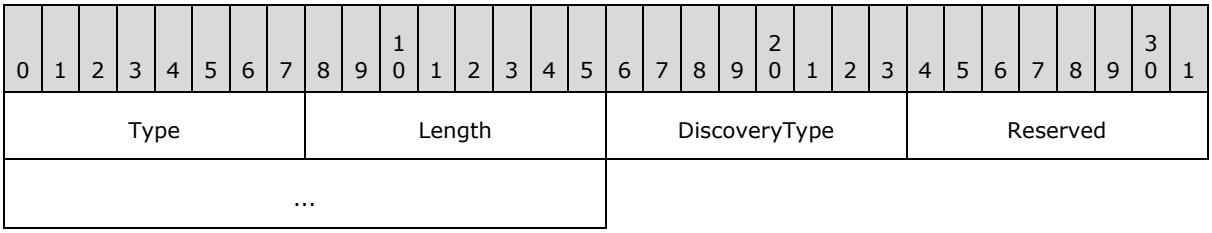


IPv4 Address (4 bytes): An IPv4 address in network byte order. This field MUST contain a valid unicast address.

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

2.2.1.3 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 ([RFC4861] section 4.3), in addition to an IPv6 Neighbor Advertisement ([RFC4861] section 4.4), which prevents packets from being relayed indirectly 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.

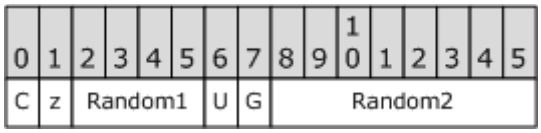
DiscoveryType (1 byte): This field MUST be set to one of the following values.

Value	Meaning
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 bubble containing **DiscoveryType** TeredoDiscoverySolicitation.

Reserved (3 bytes): MUST be set to zero when sent and MUST be ignored on receipt.

2.2.1.4 Random Address Flags

The **Flags** field as specified in [RFC4380] section 4 contains reserved bits that this extension uses as follows.



C: This flag is specified in [RFC4380].

z: This flag is reserved. It MUST be set to zero when the address is constructed, as specified in [RFC4380].

Random1: MUST be set to a random value.

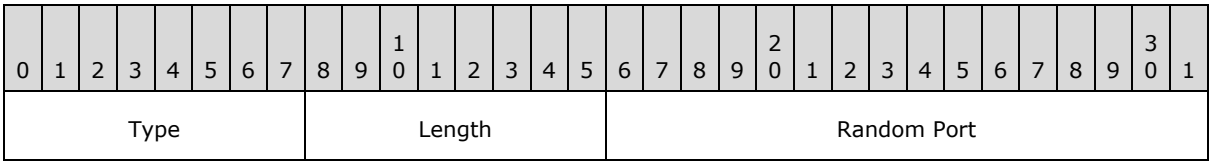
U: This flag is specified in [RFC4380].

G: This flag is specified in [RFC4380].

Random2: MUST be set to a random value.

2.2.1.5 Random Port Trailer

The Random Port Trailer is used by the Port-Preserving Symmetric NAT Extension in both indirect and direct bubbles, and by the sequential port-symmetric NAT extension in indirect bubbles.<2>



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

Length (1 byte): The length, in bytes, of the remainder 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.

3 Protocol Details

3.1 Common Details

The details common to all extensions are simply what is specified in [RFC4380]. The purpose of this section is to provide a context for implementation-specific notes about the base Teredo protocol.

3.1.1 Abstract Data Model

No state is necessary other than that specified in [RFC4380] section 5.2.

3.1.2 Timers

The Teredo protocol requires the following timers:

- The qualification timer, as specified in [RFC4380] section 5.2.1.
- The refresh interval timer, as specified in [RFC4380] section 5.2.7.

[RFC4380] section 5.2 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" ([RFC4380] section 5.2.7) 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<3> 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.

3.1.3 Initialization

The initialization rules are specified in [RFC4380].<4>

Neighbor Unreachability Detection (as specified in [RFC4861] section 7.3) SHOULD be enabled on the Teredo interface.<5> (Note that [RFC4861] section 1 explains that the portions of Neighbor Discovery that rely on link-layer multicast do not apply to link types that do not provide multicast, and Teredo is in this category, but that Neighbor Unreachability Detection is still expected to be provided as specified.)

3.1.4 Higher-Layer Triggered Events

None are required other than those specified in [RFC4380].

3.1.5 Message Processing Events and Sequencing Rules

None are required other than those specified in [RFC4380] except as follows.<6>

[RFC4380] section 5.2.1 states that "the system first performs the 'start action' by sending a Router Solicitation message, as defined in [RFC4861]. The client picks a link-local address and uses it as

the IPv6 source of the message". The section further states that a Teredo client checks whether a router advertisement (RA) is valid, where a valid RA "should contain the global Teredo IPv6 address service prefix", and that invalid ones are discarded.

This document further refines the behavior as follows. A Teredo client SHOULD source the router solicitation packets for qualification from a link local address other than fe80::5445:5245:444f or fe80::ffff:ffff:ffff:ffff to get an RA from the Teredo server that contains a global Teredo IPv6 address service prefix but MAY<7> instead choose not to, so that the Teredo client can get an RA from the Teredo server that contains an alternate Teredo IPv6 address prefix, such as the 3ffe:831f::/32 prefix.

A Teredo client SHOULD accept RAs that contain the global Teredo IPv6 service prefix but MAY<8> instead accept RAs that contain an alternate Teredo IPv6 address prefix, such as the 3ffe:831f::/32 prefix.

An implementation MAY<9> support being a Teredo server. A Teredo server SHOULD advertise the global Teredo IPv6 service prefix but MAY<10> instead advertise in RAs a private Teredo IPv6 address service prefix if the router solicitation came from a link local address of fe80::5445:5245:444f or fe80::ffff:ffff:ffff:ffff.

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. Otherwise, the Teredo client MUST skip past the trailer.

If a malformed trailer appears in the sequence of Trailers in the packet, the Teredo client MUST stop processing Trailers and ignore them (but not drop 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 defined in section 2.2.1).
- The length in bytes of the remainder of the UDP datagram is less than 2 + the value of the **Length** field of the trailer.

3.1.6 Timer Events

The following timer event processing is required by the Teredo client:

- Qualification timer processing, as specified in [RFC4380] section 5.2.1.<11>
- Refresh interval timer processing, as specified in [RFC4380] section 5.2.7.

3.1.7 Other Local Events

In addition to the events specified in [RFC4380], packets equivalent to those sent for a peer the first time a connection is being established MAY<12> be generated at other implementation-specific times.

Similarly, [RFC4380] section 5.2.5 states:

"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<13> choose to send additional router solicitation messages to the server at other implementation-specific times.

3.2 Symmetric NAT Support Extension Details

[RFC4380] section 5.2.1 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, [RFC4380] section 5.2.9 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 [RFC2463] section 4). However, the direct IPv6 connectivity test is limited only to communication between Teredo IPv6 addresses and non-Teredo IPv6 addresses. In the following extension, the use of a nonce in direct and indirect bubbles is introduced, and a mechanism to verify that the mapped address/port are not spoofed is provided.

This extension is optional; an implementation SHOULD <14> support it.

3.2.1 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 [RFC4380] section 5.2, 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.

3.2.2 Timers

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

3.2.3 Initialization

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

3.2.4 Higher-Layer Triggered Events

None are required other than those specified in [RFC4380].

3.2.5 Message Processing Events and Sequencing Rules

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

3.2.5.1 Sending an Indirect Bubble

The rules for when indirect bubbles are sent to a Teredo peer are specified in [RFC4380] section 5.2.6. 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 2.2.1.1) appended to the indirect bubble, and also store it in the **Nonce Sent** field of its **Peer Entry** for that Teredo peer.

3.2.5.2 Sending a Direct Bubble

The rules for when direct bubbles are sent to a Teredo peer are specified in [RFC4380] section 5.2.6. 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.

3.2.5.3 Receiving an Indirect Bubble

The rules for processing an indirect bubble are specified in [RFC4380] section 5.2.3. 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 nonzero, the **Nonce Received** field SHOULD<15> be set to zero.

3.2.5.4 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 [RFC4380] section 5.2.3.

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 preceding 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 [RFC4380] section 5.2.

3.2.6 Timer Events

None are required other than those in [RFC4380].

3.2.7 Other Local Events

None.

3.3 UPnP-Enabled Symmetric NAT Extension Details

The UPnP-enabled Symmetric NAT Extension is optional; an implementation SHOULD<16> support it. This extension has the Symmetric NAT Support Extension (as specified in section 3.2) as a dependency. Any node that implements this extension MUST also implement the Symmetric NAT Support Extension.

3.3.1 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 3.2.1 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.

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.

3.3.2 Timers

None are required other than those in [RFC4380].

3.3.3 Initialization

Prior to beginning the qualification procedure, the Teredo client MUST call the AddPortMapping function, as specified in [UPNPWANIP] section 2.4.16, 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. The Teredo client MUST set its **UPnP-Enabled NAT flag** based on whether the AddPortMapping function succeeded or failed.

During the qualification procedure (as specified in [RFC4380] section 5.2.1), when the Teredo client receives a response from the secondary Teredo server, the Teredo client MUST compare the mapped address/port that is learned from the secondary Teredo server with the mapped address/port that is 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 that was learned from the Teredo server MUST be compared with 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.

3.3.4 Higher-Layer Triggered Events

None are required other than those in [RFC4380].

3.3.5 Message Processing Events and Sequencing Rules

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

3.3.5.1 Receiving a Direct Bubble

Except indicated as follows, the rules for handling a direct bubble are as specified in section 3.2.5.4.

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**.

3.3.5.2 Sending a Direct Bubble

The rules for sending a direct bubble are specified in [RFC4380] section 5.2.6 and in section 3.2.5.2. 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 can 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.

3.3.5.3 Sending a Data Packet

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

If the Teredo client sending the data packet meets all the following criteria, then it MUST send the data packet to the mapped address/port embedded in the peer's Teredo IPv6 address.

- 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.

3.3.6 Timer Events

None beyond those in [RFC4380].

3.3.7 Other Local Events

None.

3.4 Port-Preserving Symmetric NAT Extension Details

The Port-Preserving Symmetric NAT Extension is optional; an implementation SHOULD<17> support it. This extension has the Symmetric NAT Support Extension (as specified in section 3.2) as a dependency. Any node that implements this extension MUST also implement the Symmetric NAT Support Extension.

3.4.1 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 3.2.1 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 communicated between the two peers.
- **Last Data Packet Sent Timestamp**: This field contains the time stamp of the last data packet sent to the peer. This time stamp is different from the field that stores the data and time of last transmission to the peer (as specified in [RFC4380] section 5.2), because the RFC-defined field is also updated every time a direct bubble is sent.

3.4.2 Timers

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

Refresh timer: A timer to refresh peer connections on which no data has been sent for a while.

3.4.3 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 refresh timer MUST be started and scheduled to expire in 30 seconds.

During the qualification procedure (as specified in [RFC4380] section 5.2.1), 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 [RFC4380] section 5.1.1) with the Local Port value. If both values match, the client MUST set the **Port-Preserving NAT** flag to TRUE.

3.4.4 Higher-Layer Triggered Events

No behavior changes are required beyond what is specified in [RFC4380] section 5.2.4, except as noted in the following.

3.4.4.1 Sending a Data Packet

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

3.4.5 Message Processing Events and Sequencing Rules

Except as specified in the following sections, the rules for message processing are as specified in section 3.2.5.

3.4.5.1 Sending an Indirect Bubble

The rules for sending an indirect bubble are as specified in section 3.2.5.1 and [RFC4380] section 5.2.6. 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 [RFC4380] section 5.2), 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. Because NAT is port-preserving, the Teredo client can predict that the assigned external port will be equal to the random port chosen, and therefore, the Teredo client MUST store the chosen port in the **Random Port** field of the **Peer Entry**.
- If the **Random Port** value is nonzero, the Teredo client MUST append a Random Port Trailer to the indirect bubble.

3.4.5.2 Sending a Direct Bubble

The rules for when direct bubbles are sent to a Teredo peer are as specified in [RFC4380] section 5.2.6. Section 3.2.5.2 also defines rules for enabling communication for clients that are positioned behind a symmetric NAT. In addition to the rules defined in the previously mentioned sections, if the **Port-Preserving NAT** flag is TRUE, the following rules will also apply:

- If the **Symmetric NAT flag** is set and the Teredo peer is not marked as "trusted" (as specified in [RFC4380] section 5.2), the Teredo client MUST send a direct bubble to the mapped address/port that is embedded in the Teredo IPv6 address of the Teredo peer.
- If the **Peer Random Port** field is nonzero, the Teredo client MUST send another direct bubble from its own random port that is destined to the **Peer Random Port**. The IPv4 destination address MUST be the mapped address that is embedded in the Teredo IPv6 address. The Teredo client MUST include the Random Port Trailer (section 2.2.1.5).

3.4.5.3 Receiving an Indirect Bubble

The rules for processing an indirect bubble are as specified in section 3.2.5.3 and [RFC4380] section 5.2.3. 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 nonzero 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 [RFC4380] section 5.2, 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 nonzero 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<18> store the new advertised port value in the **Peer Random Port** field, and if the Port-Preserving NAT Flag is TRUE, clear the **Random Port** field and stop listening on the old random port. This allows communication to be reestablished if either side changes the random port that it is using.

3.4.5.4 Receiving a Direct Bubble

The rules for handling direct bubbles are specified in section 3.2.5.4 and [RFC4380] section 5.2.3. If the **Port-Preserving NAT** flag is TRUE, the rules for whether to accept a direct bubble are extended as follows:

- 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 [RFC4380] section 5.2. 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 Port** 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 [RFC4380] section 5.2, 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 reestablished. 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 [RFC4380] section 5.2.
- If the direct bubble is received on the random port, the Teredo peer is "trusted," the **Direct Receive on Primary Port** flag is FALSE, and the **Direct Receive on Random Port** flag is set to TRUE, 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<19> 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 3.4.5.1.

3.4.6 Timer Events

3.4.6.1 Refresh Timer Expiry

When the 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 3.4.5.2) 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 refresh timer MUST then be rescheduled to expire in 30 seconds.

3.4.7 Other Local Events

None.

3.5 Sequential Port-Symmetric NAT Extension Details

The Sequential Port-Symmetric NAT Extension is optional; an implementation SHOULD<20> support it. This extension has the Symmetric NAT Support Extension (section 3.2) 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 3.4).

3.5.1 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 the behavior described in this document. The Sequential Port-Symmetric NAT Extension extends the abstract data model in section 3.1.1 by adding the following additional fields.

Peer Entry: The following fields need to be added for each peer.

- EchoTestNonce1: This field contains the value of the nonce that is sent as part of the authentication encapsulation (as specified in [RFC4380] section 5.1.1) in the router solicitation packet, which is sent to the Teredo server address as part of the **echo test**.
- EchoTestNonce2: This field contains the value of the nonce that is sent as part of the authentication encapsulation in the router solicitation packet, which is sent to the secondary Teredo server address as part of the echo test.
- EchoTestLowerPort: This field contains the value of the external port mapping that is extracted from the origin indication of the router advertisement that is 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: This field contains the value of the external port mapping that is extracted from the origin indication of the router advertisement that is received from the secondary Teredo server address as part of the echo test. A value of 0 indicates that no router advertisement has been received.
- EchoTestRetryCounter: This field contains the number of times that an echo test was attempted.

3.5.2 Timers

In addition to the timers specified in section 3.4.2, the following additional timer is required for each peer entry.

Echo Test Failover Timer: A single-use timer that runs whenever an echo test is in progress.

3.5.3 Initialization

No initialization is necessary except the initialization that is specified in section 3.4.3.

3.5.4 Higher-Layer Triggered Events

No behavior changes are required except what is specified in section 3.4.4.

3.5.5 Message Processing Events and Sequencing Rules

Except as specified in the following sections, the rules for message processing are as specified in section 3.4.5.

3.5.5.1 Handling a Request to Send an Indirect Bubble

When the Teredo service (as specified in [RFC4380]) or the Teredo Extensions specify that an indirect bubble is to be sent, the following actions apply at that time. Any behavior of the Teredo service or the Teredo Extensions still applies to how indirect bubbles are constructed; however, such behavior is done at a later time; as specified in section 3.5.5.4.

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 [RFC4380] section 5.2) 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 in Starting the Echo Test.

3.5.5.1.1 Starting the Echo Test

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

1. A router solicitation, as specified in [RFC4380] section 5.2.1, MUST be sent to the Teredo server address. The router solicitation MUST include an authentication encapsulation with a randomly-generated Nonce field (see [RFC4380] section 5.1.1). The nonce included in the authentication encapsulation MUST then be stored in the **EchoTestNonce1** field of the peer entry.

2. A direct bubble MUST be sent to the peer.

3. 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. 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** field and set the Echo Test Failover Timer to expire in a number of seconds equal to **EchoTestRetryCounter**.

3.5.5.2 Sending an Indirect Bubble

The rules for sending an indirect bubble are as specified in section 3.2.5.1 and [RFC4380] section 5.2.6. 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 nonzero, then the Teredo client MUST append a Random Port Trailer (section 2.2.1.5) to the indirect bubble.

3.5.5.3 Receiving a Direct Bubble

Regardless of the state of the **Port-Preserving NAT** flag, the processing of the direct bubble MUST be completed as specified in section 3.4.5.4. After processing is completed, if the **Direct Receive on Primary Port** flag is TRUE and the Echo Test Failover Timer is running, the Echo Test Failover Timer MUST be canceled and the **EchoTestLowerPort**, **EchoTestUpperPort**, and **EchoTestRetryCounter** fields MUST all be set to zero.

3.5.5.4 Receiving a Router Advertisement

The rules for processing a router advertisement are as specified in [RFC4380] section 5.2.1. 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, **EchoTestLowerPort** MUST be set to the external port mapping that is extracted from the origin indication of this router advertisement.
- If the received nonce is equal to **EchoTestNonce2** and **EchoTestUpperPort** is zero, **EchoTestUpperPort** MUST be set to the external port mapping that is extracted from the origin indication of this router advertisement.
- If the **EchoTestUpperPort** and **EchoTestLowerPort** are now both nonzero, 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 3.5.5.2.

3.5.6 Timer Events

3.5.6.1 Refresh Timer Expiry

The processing of the Refresh Timer Expiry MUST be completed as specified in section 3.4.6.1. In addition to those rules, the Teredo client MUST set the **EchoTestLowerPort**, **EchoTestUpperPort**, and **EchoTestRetryCounter** to zero.

3.5.6.2 Echo Test Failover Expiration

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

- If the value of the **EchoTestRetryCounter** field is two, the Teredo client MUST send an indirect bubble as specified in section 3.2.5.1.
- If the value of the **EchoTestRetryCounter** field is one, the Teredo client MUST start another echo test as specified in section 3.5.5.1.1.

3.5.7 Other Local Events

None.

3.6 Hairpinning Extension Details

This extension is optional; an implementation SHOULD<21> support it.

3.6.1 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 [RFC4380] section 5.2, 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.

3.6.2 Timers

None are required other than those specified in [RFC4380].

3.6.3 Initialization

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

Prior to beginning the qualification procedure, the Teredo client MUST call the AddPortMapping function (as specified in [UPNPWANIP] section 2.4.16) with the parameters specified in section 3.3.3. 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.

3.6.4 Higher-Layer Triggered Events

None are required other than those specified in [RFC4380] section 5.2.3.

3.6.5 Message Processing Events and Sequencing Rules

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

3.6.5.1 Sending an Indirect Bubble

The rules for when indirect bubbles are sent to a Teredo peer are as specified in [RFC4380] section 5.2.6. 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 nonzero, the Alternate Address Trailer MUST also include it in the list.

Hairpinning requires "direct IPv6 connectivity tests" (as specified in [RFC4380] section 5.2.9) 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.

3.6.5.2 Receiving an Indirect Bubble

The rules for processing indirect bubbles are as specified in [RFC4380] section 5.2.3. In addition to those rules, when a Teredo client receives an indirect bubble with the Alternate Address Trailer, it SHOULD<22> first verify that the Alternate Address Trailer is correctly formed (as specified in section 2.2.1.2), 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 [RFC4380] section 5.2.6), 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.

3.6.5.3 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 [RFC4380] section 5.2.3.

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 preceding rules matches, the direct bubble MUST be dropped.

3.6.6 Timer Events

None are required other than those specified in [RFC4380].

3.6.7 Other Local Events

None.

3.7 Server Load Reduction Extension Details

This extension is optional; an implementation SHOULD<23> support it.

3.7.1 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 [RFC4380] section 5.2, 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.

3.7.2 Timers

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

3.7.3 Initialization

None are required other than those specified in [RFC4380].

3.7.4 Higher-Layer Triggered Events

No behavior changes are required beyond what is specified in [RFC4380] section 5.2.4, except as noted in the following.

3.7.4.1 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 [RFC4380] section 5.2). 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.

3.7.5 Message Processing Events and Sequencing Rules

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

3.7.5.1 Receiving a Direct Bubble

The rules for processing direct bubbles are as specified in [RFC4380] section 5.2.3. 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.

3.7.6 Timer Events

3.7.6.1 Retransmission Timer Expiry

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 reestablish connectivity as if no communication between the Teredo client and the Teredo peer had been established.

3.7.7 Other Local Events

None.

3.8 Random Address Extension Details

This extension is strongly recommended; an implementation SHOULD<24> support it.

3.8.1 Abstract Data Model

No state is necessary beyond that specified in [RFC4380] section 5.2.

3.8.2 Timers

None are required other than those specified in [RFC4380].

3.8.3 Initialization

In addition to what is specified in [RFC4380] section 5.2.1 on constructing the Teredo IPv6 address after the qualification procedure is complete, the 12 bits specified in section 2.2.1.4 MUST be set to random values.

3.8.4 Higher-Layer Triggered Events

None are required other than those specified in [RFC4380].

3.8.5 Message Processing Events and Sequencing Rules

None are required other than those specified in [RFC4380] section 5.2.3.

3.8.6 Timer Events

None are required other than those specified in [RFC4380].

3.8.7 Other Local Events

None.

4 Protocol Examples

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

4.1 Symmetric NAT Support Extension

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

In the figure in section 1.3.1, 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 [RFC4380] section 5.2.1. 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).

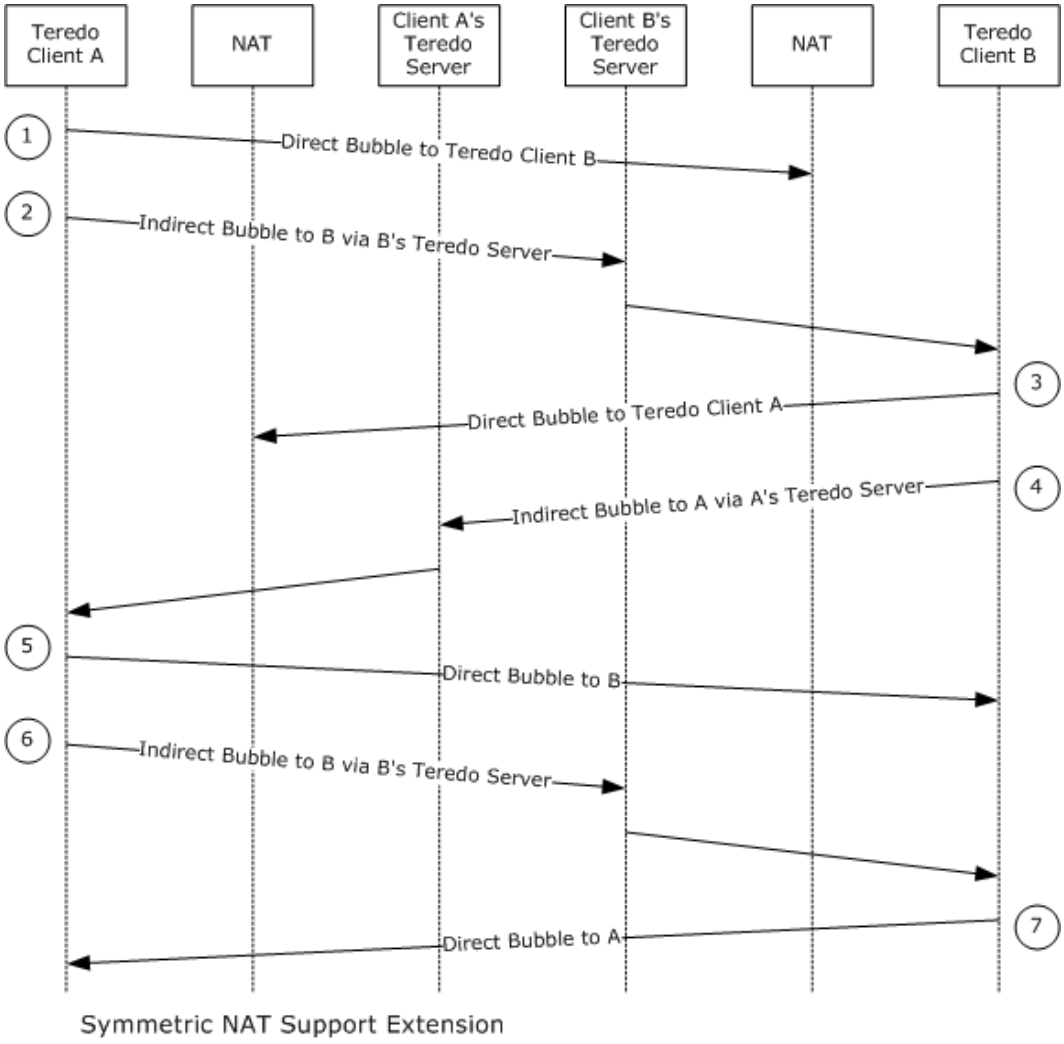


Figure 4: Port-symmetric NAT to address-restricted NAT packet exchange

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 the NAT 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 aforementioned 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 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.

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).

4.2 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).

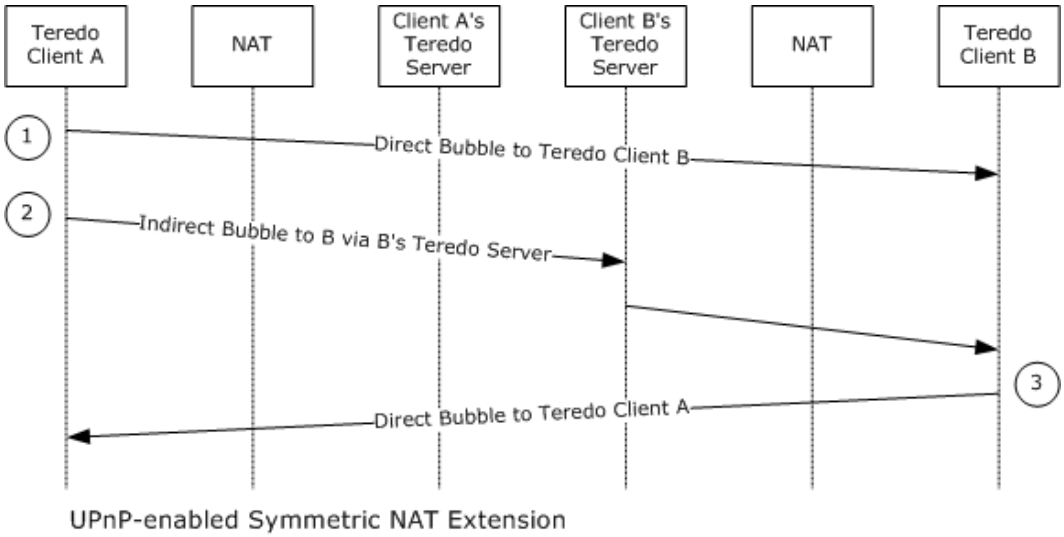


Figure 5: UPnP-enabled symmetric NAT packet exchange

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 aforementioned 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 preceding 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 aforementioned 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).

4.3 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.

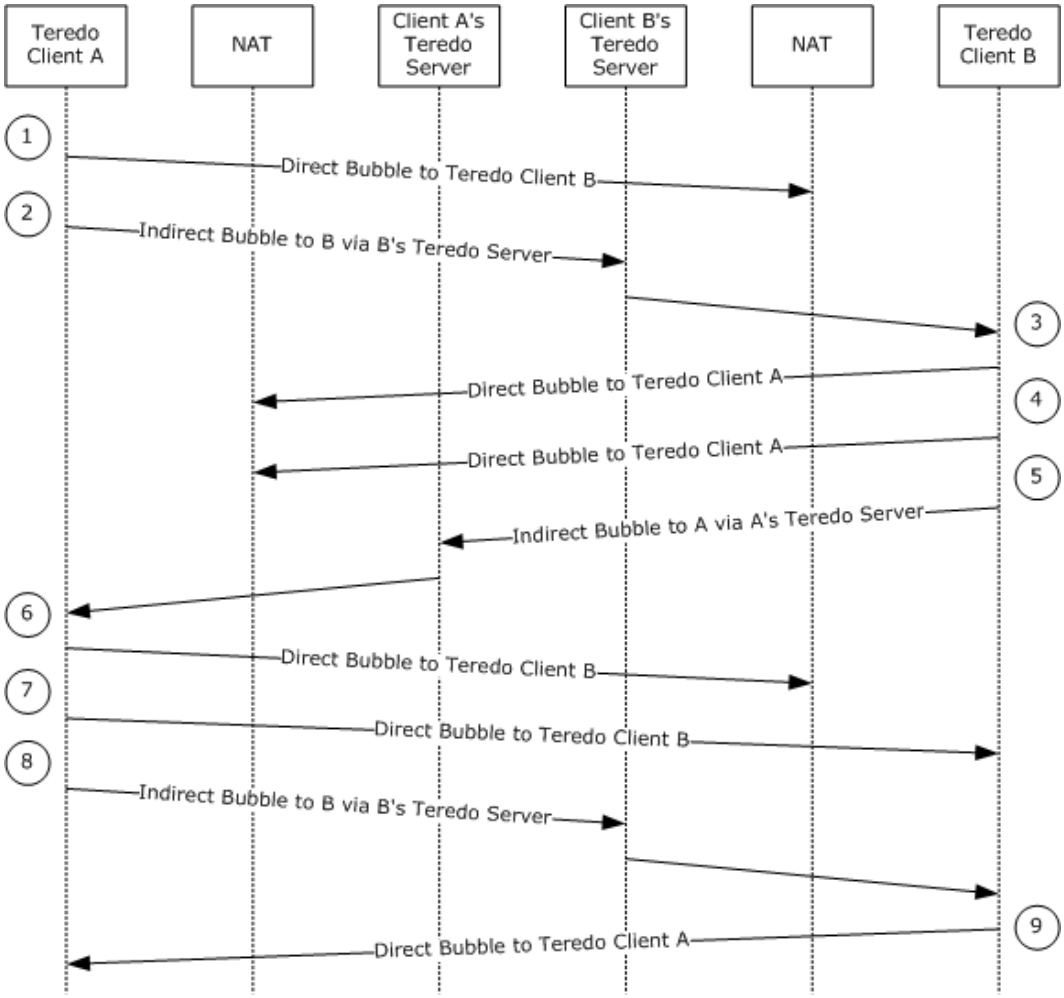


Figure 6: Port-preserving symmetric NAT packet exchange

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 nonzero. Because 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; therefore, it needs to use a random port. It checks if its **Random Port** field is set to nonzero. Because 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 preceding 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 previously.

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, communication is now possible in the other direction (A to B).

4.4 Sequential Port-Symmetric NAT Extension

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

Assume that the 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 the 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.

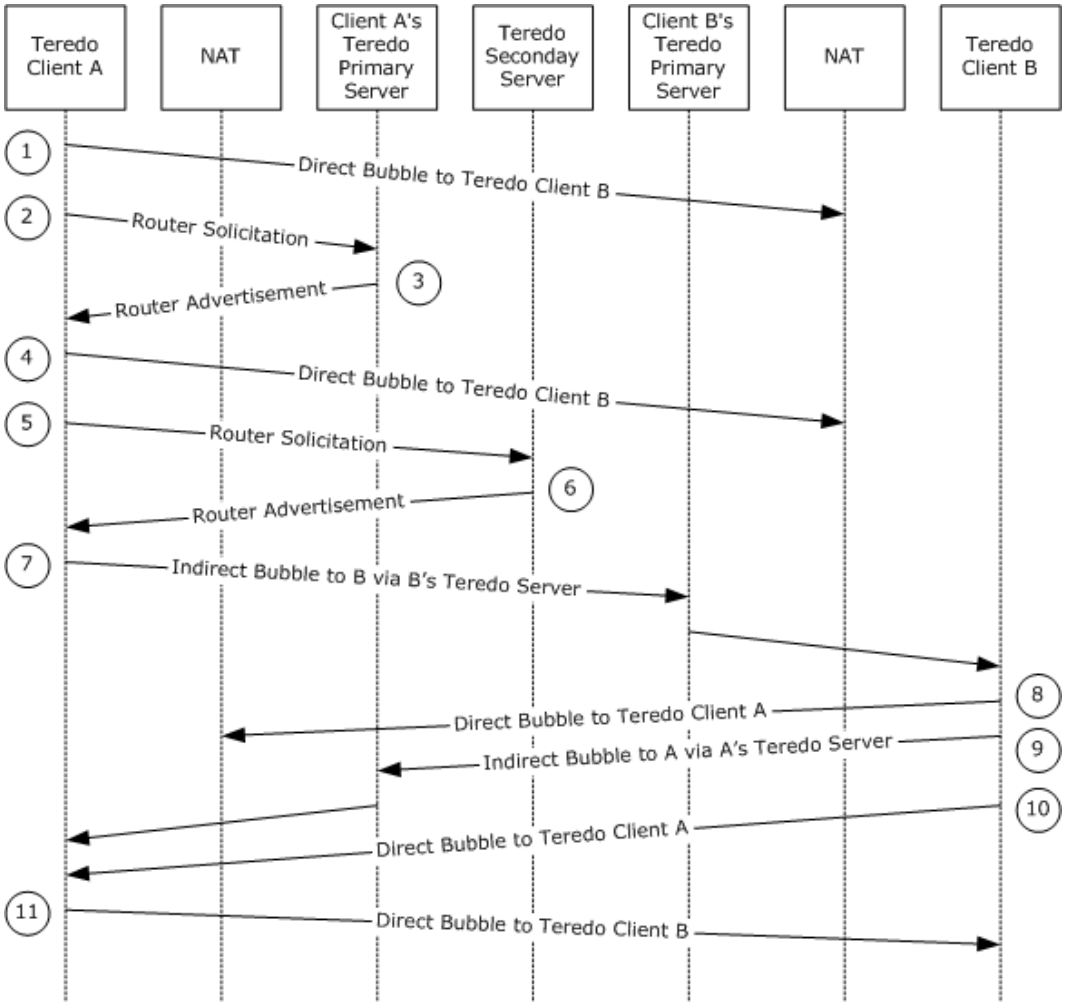


Figure 7: Sequential port-symmetric NAT packet exchange

1. During qualification, Client A will compare its outgoing port to its incoming port to determine if it is behind a port-preserving NAT. Because 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. Client 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 NAT, **address-restricted NAT**, or UPnP-enabled NAT. Because Client A is behind a symmetric NAT, the external port that is used by Client A's NAT is a new port. This direct bubble is dropped by Client B's NAT because Client B is behind a port-restricted NAT.

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

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

    1. 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).

2. 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, for example, port 2100. This direct bubble is dropped by Client B's port-preserving NAT.

3. 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).

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

5. Client A can then calculate the predicted port that is used for Packet 2 as the average (rounded down) of 1200 and 1202, that is, 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:

    1. 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.) Because in this example, A's NAT is a normal sequential port-symmetric NAT, this packet is dropped.

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

    3. The next packet is a direct bubble (Packet 10) sent to port 1201. This packet makes it in to Client A because Client A previously sent an outgoing packet (Packet 2) 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).

4.5 Hairpinning Extension and Random Address Extension

The following protocol example illustrates the use of both the Hairpinning Extension and the Random Address Extension. Each of these extensions can be independently implemented. A single scenario can be used to independently describe their implementation, as follows.

In the figure in section 1.3.5, 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 (for example, NAT E) is common to both A and B but the immediate NATs that they are connected to are different (for example, 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). It is assumed 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 [RFC4380] section 5.2.1. When configuring their respective Teredo IPv6 addresses, as part of the Random Address Extension, both clients randomly set the 12 bits, as specified in section 2.2.1.4. Thus the Teredo IPv6 address of both A and B includes the randomly set field of flags and the mapped address/port values determined as part of the qualification procedure.

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.

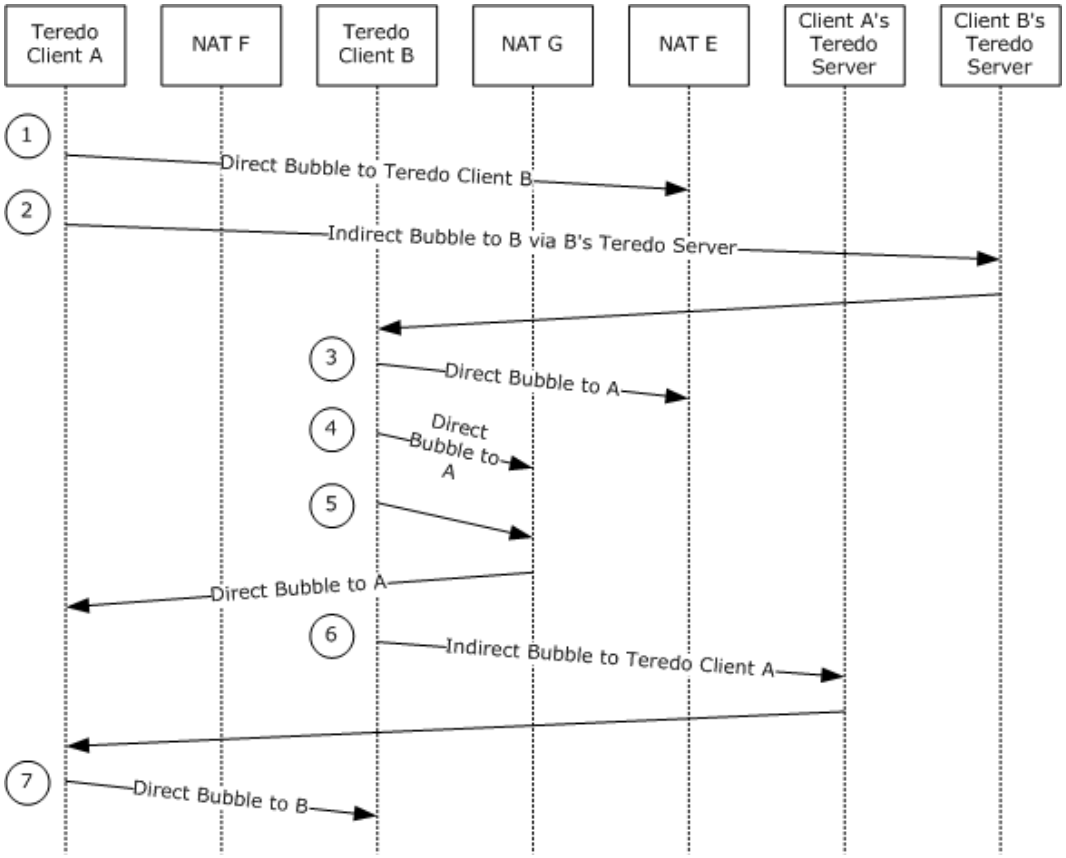


Figure 8: Hairpinning-based packet exchange

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 the 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 [RFC4380] section 5.2. Also, the mapped address status field (as specified in [RFC4380] section 5.2.3) is changed to "trusted." At this point, communication in one direction is now possible (A to B, but not vice versa).

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).

4.6 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 reestablished because A wants to send a data packet to B.

The following packet exchange helps A reestablish communication with B.

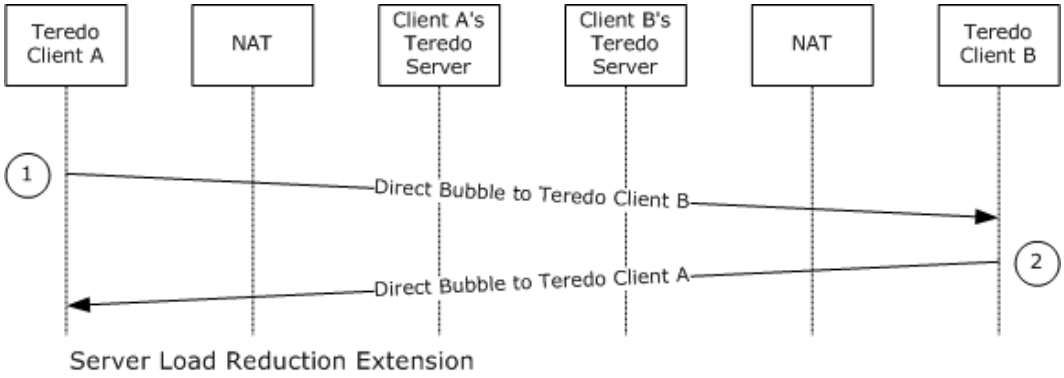


Figure 9: Server load reduction packet exchange

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.

5 Security

The following sections specify security considerations for implementers and an index of security parameters for Teredo Extensions.

5.1 Security Considerations for Implementers

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

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 2.2.1.2 explicitly limits the number of addresses that a Teredo client will accept. Because the Teredo service punches a hole in the NAT to be able to talk to the Teredo server, attackers can use this channel to reach an internal client machine. The Random Address Extension specified in this document obscures the Teredo address to a greater extent than [RFC4380] section 7 does, making it more difficult for attackers to send data packets to the internal clients.

Because the nonce in the Nonce Trailer is used (as specified in section 3.2.5.4) 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.

The Teredo IPv6 Address format defined in [RFC4380] section 4 makes it relatively easy for a malicious user to conduct an address-scan to determine IPv6 addresses by guessing the external (mapped) IPv4 address and port assigned to the Teredo client. The Random Address Extension guards against address-scanning risks by providing a range of 4096 IPv6 addresses per external IPv4 address/port. As a result, even if a malicious user were able to determine the external (mapped) IPv4 address and port assigned to the Teredo client, the malicious user would still need to attack a range of 4,096 IPv6 addresses to determine the actual Teredo IPv6 address of the client.

5.2 Index of Security Parameters

Security parameter	Section
Nonce	2.2.1.1
Random Address Flags	2.2.1.4

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows® XP operating system Service Pack 2 (SP2)
- Windows Server® 2003 operating system with Service Pack 1 (SP1)
- Windows Vista® operating system
- Windows Server® 2008 operating system
- Windows® 7 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 1.5: Windows resolves the Teredo server name by using DNS to determine the IPv4 address of the server. Section 5.2 of [RFC4380] requires the Teredo client to be configured with the two addresses of a Teredo server: the primary Teredo server address and the secondary Teredo server address. Windows assumes that the IPv4 addresses that are returned by DNS contain only the primary IPv4 addresses of Teredo servers. The secondary Teredo server address is deduced by adding one to the value of the primary Teredo server address.

<2> Section 2.2.1.5: The sequential port-symmetric NAT extension is supported in Windows Server 2008 R2 only.

<3> Section 3.1.2: Windows Server 2003 with SP1 sets the randomized refresh interval equal to the refresh interval. Windows XP SP2 and Windows XP SP3 set the randomized refresh interval to a value chosen between 50% and 150% of the refresh interval. Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 set the randomized refresh interval to a value chosen between 100% and 150% of the refresh interval.

<4> Section 3.1.3: The qualification procedure is specified in [RFC4380] section 5.2.1. Currently Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2 completely skip the first phase of the qualification procedure and implement only the second phase where it uses the Teredo link local address with the cone bit set to zero. Consequently, a distinction between cone and restricted NATs can no longer be made. Teredo communication will still succeed, but at the expense of forcing peers to skip step 4 of the sending details specified in [RFC4380]. This will result in the same number of indirect bubbles being sent as if the other end were a peer behind a restricted NAT. Even though the peer behind the cone NAT does not need these indirect bubbles, it replies to these indirect bubbles just like it would to any other indirect bubbles. Skipping step 4 is already allowed for reliability reasons (as specified in [RFC4380] section 5.2.4), and hence this does not break interoperability, but the result of skipping the first phase of qualification is to force that behavior (which is less efficient, but potentially more reliable) to be taken by peers. Windows XP SP2,

Windows XP SP3, and Windows Server 2003 with SP1 implementations do not skip phase 1 and hence are able to differentiate between cone and restricted NATs.

<5> Section 3.1.3: Windows does not support Neighbor Unreachability Detection on Teredo interfaces.

<6> Section 3.1.5: Clients may ignore the cone bit and treat it as if it were always clear, as specified in [RFC4380] section 5.2.4 (last paragraph). Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2 do this. Windows XP SP2, Windows XP SP3, and Windows Server 2003 with SP1 implementations do not ignore the cone bit. Also, [RFC4380] section 5.2.4 specifies that packets will be de-queued and forwarded when the client receives a bubble or another packet directly from this Teredo peer. Windows will de-queue and forward the packet when the client receives an indirect bubble before a direct bubble from this peer, and as a result, the packet may be dropped in the network if no direct bubble from the peer has yet opened the port in the peer's NAT. This is why [RFC4380] specifies to wait until the direct bubble is received, marking the peer's entry as "trusted".

<7> Section 3.1.5: Windows XP SP2 (but not later service packs) and Windows Server 2003 with SP1 Teredo clients produce router solicitation packets sourced from fe80:::5445:5245:444f.

<8> Section 3.1.5: Windows XP SP2 (but not later service packs) and Windows Server 2003 with SP1 only accept RAs that contain a 3FFE:831F::/32 prefix instead of the 2001:0000::/32 prefix.

<9> Section 3.1.5: Windows Server 2008 R2 supports being configured as a Teredo server.

<10> Section 3.1.5: Windows Server 2008 R2, when configured as a Teredo server, will advertise the 3FFE:831F::/32 prefix instead of the 2001:0000::/32 prefix, if the router solicitation packet is sourced from fe80::5445:5245:444f or fe80::ffff:ffff:ffff:ffff.

<11> Section 3.1.6: [RFC4380] section 5.2.1 also specifies a "default" timeout but does not mandate any particular algorithm. In Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2, the timeout value is started from 1 second for every qualification procedure and is doubled each time a timeout is triggered, up to a maximum of six timeouts before giving up. Windows XP SP2, Windows XP SP3, and Windows Server 2003 with SP1 implementations use the "default" timeout values, as specified in [RFC4380] section 5.2.1.

<12> Section 3.1.7: When the Neighbor Cache entry (as specified in [RFC4861] section 5.1) of the peer would have entered the PROBE state if the Teredo interface supported Neighbor Unreachability Detection with a Reachable Time (as specified in [RFC4861] section 4.2) of 15 seconds, Windows generates packets, excluding indirect bubbles, equivalent to those sent for a peer the first time a connection is being established.

In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, if Teredo qualification takes greater than 2 seconds, and if during this qualification, attempts to connect to a new peer are being made, then after the Teredo client has qualified it will concurrently transmit the bubbles that are sent to the peer to establish a connection.

In Windows 7 and Windows Server 2008 R2, if the Echo Test specified in section 3.5.5.1.1 was performed, then the Teredo client sends to the peer an extra direct bubble, identical to that generated during the Echo Test. The identical direct bubble is sent after the indirect bubble, specified in section 3.5.5.2, is sent to the peer.

Windows Vista SP1, Windows Server 2008, Windows 7, and Windows Server 2008 R2 send an extra (unnecessary) direct bubble to each mapped address/port advertised in the Alternate Address Trailer if the status of the peer is "not-trusted" and the **Peer Random Port** field of the **Peer Entry** is nonzero. This extra set of direct bubbles is sent on receiving an indirect bubble, as specified in section 3.6.5.2.

<13> Section 3.1.7: Windows Vista, Windows Vista SP1, Windows Server 2008, Windows 7, and Windows Server 2008 R2 send an additional router solicitation message to the server once the Teredo client is qualified and when the Teredo client is used for the first time by an application.

<14> Section 3.2: Windows XP SP2, Windows XP SP3, and Windows Server 2003 with SP1 do not support the Symmetric NAT Support Extension.

<15> Section 3.2.5.3: Windows Server 2008, Windows Server 2008 R2, Windows Vista, and Windows 7 do not clear the **Nonce Received** field when receiving an indirect bubble with no Nonce Trailer.

<16> Section 3.3: Windows XP SP2, Windows XP SP3, and Windows Server 2003 with SP1 do not support the UPnP-enabled Symmetric NAT Extension.

<17> Section 3.4: Windows XP SP2, Windows XP SP3, Windows Server 2003 with SP1, and Windows Vista do not support the Port-Preserving Symmetric NAT Extension.

<18> Section 3.4.5.3: Windows Server 2008 does accept the new **Peer Random Port**, but it currently does not stop using its own old random port.

<19> Section 3.4.5.4: Windows Server 2008 does not accept the new port mapping learned from the incoming direct bubble. Instead it ignores the direct bubble.

<20> Section 3.5: Windows XP SP2, Windows XP SP3, Windows Server 2003 with SP1, Windows Vista, and Windows Server 2008 do not support the Sequential Port-Symmetric NAT Extension.

<21> Section 3.6: Windows XP SP2, Windows XP SP3, Windows Server 2003 with SP1, and Windows Vista RTM do not support the Hairpinning Extension.

<22> Section 3.6.5.2: Windows Vista SP1, Windows 7, Windows Server 2008, and Windows Server 2008 R2 will accept a trailer that includes more than four alternate address/port pairs, but will only use the first four.

<23> Section 3.7: Windows does not support the Server Load Reduction Extension.

<24> Section 3.8: Windows XP SP2, Windows XP SP3, Windows Server 2003 with SP1, and Windows Vista RTM do not support the Random Address Extension.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

L

M

N

O

P

R

S

T

U

V