Internet Engineering Task Force (IETF)
Request for Comments: 6145
Obsoletes: 2765
Category: Standards Track
ISSN: 2070-1721

X. Li
C. Bao
CERNET Center/Tsinghua University
F. Baker
Cisco Systems

April 2011

IP/ICMP Translation Algorithm

Abstract

This document describes the Stateless IP/ICMP Translation Algorithm (SIIT), which translates between IPv4 and IPv6 packet headers (including ICMP headers). This document obsoletes RFC 2765.

Status of This Memo

This is an Internet Standards Track document.

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

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6145.

Copyright Notice

Table of Contents

1. Introduction and Motivation

This document is a product of the 2008-2010 effort to define a replacement for NAT-PT [RFC2766] (which was changed to Historic status when [RFC4966] was published in 2007). It is directly derived from Erik Nordmark's "Stateless IP/ICMP Translation Algorithm (SIIT)" [RFC2765], which provides stateless translation between IPv4 [RFC0791] and IPv6 [RFC2460], and between ICMPv4 [RFC0792] and ICMPv6 [RFC4443]. This document obsoletes RFC 2765 [RFC2765]. The changes from RFC 2765 [RFC2765] are listed in Section 2.

Readers of this document are expected to have read and understood the framework described in [RFC6144]. Implementations of this IPv4/IPv6 translation specification MUST also support the address translation algorithms in [RFC6052]. Implementations MAY also support stateful translation [RFC6146].

1.1. IPv4-IPv6 Translation Model

The translation model consists of two or more network domains connected by one or more IP/ICMP translators (XLATs) as shown in Figure 1.

```
                ---------          ---------
              //        \\       //         \\
            /             +----+              \
           |              |XLAT|               | XLAT: IP/ICMP
           |   IPv4       +----+   IPv6        |       Translator
           |   Domain     |    |   Domain      |
           |              |    |               |
            \             |    |              /
             \\         //      \\          //
                --------          ---------
```

Figure 1: IPv4-IPv6 Translation Model

The scenarios of the translation model are discussed in [RFC6144].

1.2. Applicability and Limitations

This document specifies the translation algorithms between IPv4 packets and IPv6 packets.

As with [RFC2765], the translating function specified in this document does not translate any IPv4 options, and it does not translate IPv6 extension headers except the Fragment Header.

The issues and algorithms in the translation of datagrams containing TCP segments are described in [RFC5382].

Fragmented IPv4 UDP packets that do not contain a UDP checksum (i.e., the UDP checksum field is zero) are not of significant use in the Internet, and in general will not be translated by the IP/ICMP translator. However, when the translator is configured to forward the packet without a UDP checksum, the fragmented IPv4 UDP packets will be translated.

Fragmented ICMP/ICMPv6 packets will not be translated by the IP/ICMP translator.

The IP/ICMP header translation specified in this document is consistent with requirements of multicast IP/ICMP headers. However, IPv4 multicast addresses [RFC5771] cannot be mapped to IPv6 multicast addresses [RFC3307] based on the unicast mapping rule [RFC6052].

1.3. Stateless vs. Stateful Mode

An IP/ICMP translator has two possible modes of operation: stateless and stateful [RFC6144]. In both cases, we assume that a system (a node or an application) that has an IPv4 address but not an IPv6 address is communicating with a system that has an IPv6 address but no IPv4 address, or that the two systems do not have contiguous routing connectivity and hence are forced to have their communications translated.

In the stateless mode, a specific IPv6 address range will represent IPv4 systems (IPv4-converted addresses), and the IPv6 systems have addresses (IPv4-translatable addresses) that can be algorithmically mapped to a subset of the service provider's IPv4 addresses. Note that IPv4-translatable addresses are a subset of IPv4-converted addresses. In general, there is no need to concern oneself with translation tables, as the IPv4 and IPv6 counterparts are algorithmically related.

In the stateful mode, a specific IPv6 address range will represent IPv4 systems (IPv4-converted addresses), but the IPv6 systems may use any IPv6 addresses [RFC4291] except in that range. In this case, a translation table is required to bind the IPv6 systems' addresses to the IPv4 addresses maintained in the translator.

The address translation mechanisms for the stateless and the stateful translations are defined in [RFC6052].

1.4. Path MTU Discovery and Fragmentation

Due to the different sizes of the IPv4 and IPv6 header, which are 20+ octets and 40 octets respectively, handling the maximum packet size is critical for the operation of the IPv4/IPv6 translator. There are three mechanisms to handle this issue: path MTU discovery (PMTUD), fragmentation, and transport-layer negotiation such as the TCP Maximum Segment Size (MSS) option [RFC0879]. Note that the translator MUST behave as a router, i.e., the translator MUST send a Packet Too Big error message or fragment the packet when the packet size exceeds the MTU of the next-hop interface.

Don’t Fragment, ICMP Packet Too Big, and packet fragmentation are discussed in Sections 4 and 5 of this document. The reassembling of fragmented packets in the stateful translator is discussed in [RFC6146], since it requires state maintenance in the translator.

2. Changes from RFC 2765

The changes from RFC 2765 are the following:

1. Redescribing the network model to map to present and projected usage. The scenarios, applicability, and limitations originally presented in RFC 2765 [RFC2765] are moved to the framework document [RFC6144].

2. Moving the address format to the address format document [RFC6052], to coordinate with other documents on the topic.

3. Describing the header translation for the stateless and stateful operations. The details of the session database and mapping table handling of the stateful translation is in the stateful translation document [RFC6146].

4. Having refined the header translation, fragmentation handling, ICMP translation and ICMP error translation in the IPv4-to-IPv6 direction, as well as the IPv6-to-IPv4 direction.

5. Adding more discussion on transport-layer header translation.

6. Adding Section 5.1.1 for "IPv6 Fragment Processing".

7. Adding Section 6 for "Special Considerations for ICMPv6 Packet Too Big".

8. Updating Section 7 for "Security Considerations".

9. Adding Appendix A "Stateless translation workflow example".

3. Conventions

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

4. Translating from IPv4 to IPv6

When an IP/ICMP translator receives an IPv4 datagram addressed to a destination towards the IPv6 domain, it translates the IPv4 header of that packet into an IPv6 header. The original IPv4 header on the packet is removed and replaced by an IPv6 header, and the transport checksum is updated as needed, if that transport is supported by the translator. The data portion of the packet is left unchanged. The IP/ICMP translator then forwards the packet based on the IPv6 destination address.

```
+-------------+                 +-------------+
|    IPv4     |                 |    IPv6     |
|   Header    |                 |   Header    |
+-------------+                 +-------------+
|  Transport- |                 |  Fragment   |
|   Layer     |      ===>       |   Header    |
|   Header    |                 | (if needed) |
+-------------+                 +-------------+
|             |                 |  Transport- |
~    Data     ~                 |   Layer     |
|             |                 |   Header    |
+-------------+                 +-------------+
                                |             |
                                ~    Data     ~
                                |             |
                                +-------------+
```

Figure 2: IPv4-to-IPv6 Translation

Path MTU discovery is mandatory in IPv6, but it is optional in IPv4. IPv6 routers never fragment a packet -- only the sender can do fragmentation.

When an IPv4 node performs path MTU discovery (by setting the Don't Fragment (DF) bit in the header), path MTU discovery can operate end-to-end, i.e., across the translator. In this case, either IPv4 or IPv6 routers (including the translator) might send back ICMP Packet Too Big messages to the sender. When the IPv6 routers send these ICMPv6 errors, they will pass through a translator that will translate the ICMPv6 error to a form that the IPv4 sender can understand. As a result, an IPv6 Fragment Header is only included if the IPv4 packet is already fragmented.

However, when the IPv4 sender does not set the DF bit, the translator MUST ensure that the packet does not exceed the path MTU on the IPv6 side. This is done by fragmenting the IPv4 packet (with Fragment Headers) so that it fits in 1280-byte IPv6 packets, since that is the minimum IPv6 MTU. The IPv6 Fragment Header has been shown to cause operational difficulties in practice due to limited firewall fragmentation support, etc. In an environment where the network owned/operated by the same entity that owns/operates the translator, the translator MAY provide a configuration function for the network administrator to adjust the threshold of the minimum IPv6 MTU to a value that reflects the real value of the minimum IPv6 MTU in the network (greater than 1280 bytes). This will help reduce the chance of including the Fragment Header in the packets.

When the IPv4 sender does not set the DF bit, the translator SHOULD always include an IPv6 Fragment Header to indicate that the sender allows fragmentation. The translator MAY provide a configuration function that allows the translator not to include the Fragment Header for the non-fragmented IPv6 packets.

The rules in Section 4.1 ensure that when packets are fragmented, either by the sender or by IPv4 routers, the low-order 16 bits of the fragment identification are carried end-to-end, ensuring that packets are correctly reassembled. In addition, the rules in Section 4.1 use the presence of an IPv6 Fragment Header to indicate that the sender might not be using path MTU discovery (i.e., the packet should not have the DF flag set should it later be translated back to IPv4).

Other than the special rules for handling fragments and path MTU discovery, the actual translation of the packet header consists of a simple translation as defined below. Note that ICMPv4 packets require special handling in order to translate the content of ICMPv4 error messages and also to add the ICMPv6 pseudo-header checksum.

The translator SHOULD make sure that the packets belonging to the same flow leave the translator in the same order in which they arrived.

4.1. Translating IPv4 Headers into IPv6 Headers

If the DF flag is not set and the IPv4 packet will result in an IPv6 packet larger than 1280 bytes, the packet SHOULD be fragmented so the resulting IPv6 packet (with Fragment Header added to each fragment) will be less than or equal to 1280 bytes. For example, if the packet is fragmented prior to the translation, the IPv4 packets should be fragmented so that their length, excluding the IPv4 header, is at most 1232 bytes (1280 minus 40 for the IPv6 header and 8 for the Fragment Header). The translator MAY provide a configuration function for the network administrator to adjust the threshold of the minimum IPv6 MTU to a value greater than 1280-byte if the real value of the minimum IPv6 MTU in the network is known to the administrator. The resulting fragments are then translated independently using the logic described below.

If the DF bit is set and the MTU of the next-hop interface is less than the total length value of the IPv4 packet plus 20, the translator MUST send an ICMPv4 "Fragmentation Needed" error message to the IPv4 source address.

If the DF bit is set and the packet is not a fragment (i.e., the More Fragments (MF) flag is not set and the Fragment Offset is equal to zero), then the translator SHOULD NOT add a Fragment Header to the resulting packet. The IPv6 header fields are set as follows:

Version: 6

Traffic Class: By default, copied from the IP Type Of Service (TOS) octet. According to [RFC2474], the semantics of the bits are identical in IPv4 and IPv6. However, in some IPv4 environments these fields might be used with the old semantics of "Type Of Service and Precedence". An implementation of a translator SHOULD support an administratively configurable option to ignore the IPv4 TOS and always set the IPv6 traffic class (TC) to zero. In addition, if the translator is at an administrative boundary, the filtering and update considerations of [RFC2475] may be applicable.

Flow Label: 0 (all zero bits)

Payload Length: Total length value from the IPv4 header, minus the size of the IPv4 header and IPv4 options, if present.

Next Header: For ICMPv4 (1), it is changed to ICMPv6 (58); otherwise, the protocol field MUST be copied from the IPv4 header.

Hop Limit: The hop limit is derived from the TTL value in the IPv4 header. Since the translator is a router, as part of forwarding the packet it needs to decrement either the IPv4 TTL (before the translation) or the IPv6 Hop Limit (after the translation). As part of decrementing the TTL or Hop Limit, the translator (as any router) MUST check for zero and send the ICMPv4 "TTL Exceeded" or ICMPv6 "Hop Limit Exceeded" error.

Source Address: The IPv4-converted address derived from the IPv4 source address per [RFC6052], Section 2.3.

If the translator gets an illegal source address (e.g., 0.0.0.0, 127.0.0.1, etc.), the translator SHOULD silently drop the packet (as discussed in Section 5.3.7 of [RFC1812]).

Destination Address: In the stateless mode, which is to say that if the IPv4 destination address is within a range of configured IPv4 stateless translation prefix, the IPv6 destination address is the IPv4-translatable address derived from the IPv4 destination address per [RFC6052], Section 2.3. A workflow example of stateless translation is shown in Appendix A of this document.

In the stateful mode (which is to say that if the IPv4 destination address is not within the range of any configured IPv4 stateless translation prefix), the IPv6 destination address and corresponding transport-layer destination port are derived from the Binding Information Bases (BIBs) reflecting current session state in the translator as described in [RFC6146].

If any IPv4 options are present in the IPv4 packet, they MUST be ignored and the packet translated normally; there is no attempt to translate the options. However, if an unexpired source route option is present then the packet MUST instead be discarded, and an ICMPv4 "Destination Unreachable, Source Route Failed" (Type 3, Code 5) error message SHOULD be returned to the sender.

If there is a need to add a Fragment Header (the DF bit is not set or the packet is a fragment), the header fields are set as above with the following exceptions:

IPv6 fields:

Payload Length: Total length value from the IPv4 header, plus 8 for the Fragment Header, minus the size of the IPv4 header and IPv4 options, if present.

Next Header: Fragment Header (44).

Fragment Header fields:

Next Header: For ICMPv4 (1), it is changed to ICMPv6 (58); otherwise, the protocol field MUST be copied from the IPv4 header.

Fragment Offset: Fragment Offset copied from the IPv4 header.

M flag: More Fragments bit copied from the IPv4 header.

Identification: The low-order 16 bits copied from the Identification field in the IPv4 header. The high-order 16 bits set to zero.

4.2. Translating ICMPv4 Headers into ICMPv6 Headers

All ICMPv4 messages that are to be translated require that the ICMPv6 checksum field be calculated as part of the translation since ICMPv6, unlike ICMPv4, has a pseudo-header checksum just like UDP and TCP.

In addition, all ICMPv4 packets MUST have the Type translated and, for ICMPv4 error messages, the included IP header also MUST be translated.

The actions needed to translate various ICMPv4 messages are as follows:

ICMPv4 query messages:

Echo and Echo Reply (Type 8 and Type 0): Adjust the Type values to 128 and 129, respectively, and adjust the ICMP checksum both to take the type change into account and to include the ICMPv6 pseudo-header.

Information Request/Reply (Type 15 and Type 16): Obsoleted in ICMPv6. Silently drop.

Timestamp and Timestamp Reply (Type 13 and Type 14): Obsoleted in ICMPv6. Silently drop.

Address Mask Request/Reply (Type 17 and Type 18): Obsoleted in ICMPv6. Silently drop.

ICMP Router Advertisement (Type 9): Single-hop message. Silently drop.

ICMP Router Solicitation (Type 10): Single-hop message. Silently drop.

Unknown ICMPv4 types: Silently drop.

IGMP messages: While the Multicast Listener Discovery (MLD) messages [RFC2710] [RFC3590] [RFC3810] are the logical IPv6 counterparts for the IPv4 IGMP messages, all the "normal" IGMP messages are single-hop messages and SHOULD be silently dropped by the translator. Other IGMP messages might be used by multicast routing protocols and, since it would be a configuration error to try to have router adjacencies across IP/ICMP translators, those packets SHOULD also be silently dropped.

ICMPv4 error messages:

Destination Unreachable (Type 3): Translate the Code as described below, set the Type to 1, and adjust the ICMP checksum both to take the type/code change into account and to include the ICMPv6 pseudo-header.

Translate the Code as follows:

Code 0, 1 (Net Unreachable, Host Unreachable): Set the Code to 0 (No route to destination).

Code 2 (Protocol Unreachable): Translate to an ICMPv6 Parameter Problem (Type 4, Code 1) and make the Pointer point to the IPv6 Next Header field.

Code 3 (Port Unreachable): Set the Code to 4 (Port unreachable).

Code 4 (Fragmentation Needed and DF was Set): Translate to an ICMPv6 Packet Too Big message (Type 2) with Code set to 0. The MTU field MUST be adjusted for the difference between the IPv4 and IPv6 header sizes, i.e., minimum(advertised MTU+20, MTU_of_IPv6_nexthop, (MTU_of_IPv4_nexthop)+20). Note that if the IPv4 router set the MTU field to zero, i.e., the router does not implement [RFC1191], then the translator MUST use the plateau values specified in [RFC1191] to determine a likely path MTU and include that path MTU in the ICMPv6 packet. (Use the greatest plateau value that is less than the returned Total Length field.)

See also the requirements in Section 6.

Code 5 (Source Route Failed): Set the Code to 0 (No route to destination). Note that this error is unlikely since source routes are not translated.

Code 6, 7, 8: Set the Code to 0 (No route to destination).

Code 9, 10 (Communication with Destination Host Administratively Prohibited): Set the Code to 1 (Communication with destination administratively prohibited).

Code 11, 12: Set the Code to 0 (No route to destination).

Code 13 (Communication Administratively Prohibited): Set the Code to 1 (Communication with destination administratively prohibited).

Code 14 (Host Precedence Violation): Silently drop.

Code 15 (Precedence cutoff in effect): Set the Code to 1 (Communication with destination administratively prohibited).

Other Code values: Silently drop.

Redirect (Type 5): Single-hop message. Silently drop.

Alternative Host Address (Type 6): Silently drop.

Source Quench (Type 4): Obsoleted in ICMPv6. Silently drop.

Time Exceeded (Type 11): Set the Type to 3, and adjust the ICMP checksum both to take the type change into account and to include the ICMPv6 pseudo-header. The Code is unchanged.

Parameter Problem (Type 12): Set the Type to 4, and adjust the ICMP checksum both to take the type/code change into account and to include the ICMPv6 pseudo-header.

Translate the Code as follows:

Code 0 (Pointer indicates the error): Set the Code to 0 (Erroneous header field encountered) and update the pointer as defined in Figure 3. (If the Original IPv4 Pointer Value is not listed or the Translated IPv6 Pointer Value is listed as "n/a", silently drop the packet.)

Code 1 (Missing a required option): Silently drop.

Code 2 (Bad length): Set the Code to 0 (Erroneous header field encountered) and update the pointer as defined in Figure 3. (If the Original IPv4 Pointer Value is not listed or the Translated IPv6 Pointer Value is listed as "n/a", silently drop the packet.)

Other Code values: Silently drop.

Unknown ICMPv4 types: Silently drop.

Original IPv4 Pointer Value		Translated IPv6 Pointer Value	
0	Version/IHL	0	Version/Traffic Class
1	Type Of Service	1	Traffic Class/Flow Label
2,3	Total Length	4	Payload Length
4,5	Identification	n/a	
6	Flags/Fragment Offset	n/a	
7	Fragment Offset	n/a	
8	Time to Live	7	Hop Limit
9	Protocol	6	Next Header
10,11	Header Checksum	n/a	
12-15	Source Address	8	Source Address
16-19	Destination Address	24	Destination Address

Figure 3: Pointer Value for Translating from IPv4 to IPv6

ICMP Error Payload: If the received ICMPv4 packet contains an ICMPv4 Extension [RFC4884], the translation of the ICMPv4 packet will cause the ICMPv6 packet to change length. When this occurs, the ICMPv6 Extension length attribute MUST be adjusted accordingly (e.g., longer due to the translation from IPv4 to IPv6). If the ICMPv4 Extension exceeds the maximum size of an ICMPv6 message on the outgoing interface, the ICMPv4 extension SHOULD be simply truncated. For extensions not defined in [RFC4884], the translator passes the extensions as opaque bit strings, and those containing IPv4 address literals will not have those addresses translated to IPv6 address literals; this may cause problems with processing of those ICMP extensions.

4.3. Translating ICMPv4 Error Messages into ICMPv6

There are some differences between the ICMPv4 and the ICMPv6 error message formats as detailed above. The ICMP error messages containing the packet in error MUST be translated just like a normal IP packet. If the translation of this "packet in error" changes the length of the datagram, the Total Length field in the outer IPv6 header MUST be updated.

```
                +-------------+             +-------------+
                |    IPv4     |             |    IPv6     |
                |   Header    |             |   Header    |
                +-------------+             +-------------+
                |   ICMPv4    |             |   ICMPv6    |
                |   Header    |             |   Header    |
                +-------------+             +-------------+
                |    IPv4     |    ===>     |    IPv6     |
                |   Header    |             |   Header    |
                +-------------+             +-------------+
                |   Partial   |             |   Partial   |
                |  Transport- |             |  Transport- |
                |   Layer     |             |   Layer     |
                |   Header    |             |   Header    |
                +-------------+             +-------------+
```

Figure 4: IPv4-to-IPv6 ICMP Error Translation

The translation of the inner IP header can be done by invoking the function that translated the outer IP headers. This process MUST stop at the first embedded header and drop the packet if it contains more embedded headers.

4.4. Generation of ICMPv4 Error Message

If the IPv4 packet is discarded, then the translator SHOULD be able to send back an ICMPv4 error message to the original sender of the packet, unless the discarded packet is itself an ICMPv4 message. The ICMPv4 message, if sent, has a Type of 3 (Destination Unreachable) and a Code of 13 (Communication Administratively Prohibited), unless otherwise specified in this document or in [RFC6146]. The translator SHOULD allow an administrator to configure whether the ICMPv4 error messages are sent, rate-limited, or not sent.

4.5. Transport-Layer Header Translation

If the address translation algorithm is not checksum neutral (see Section 4.1 of [RFC6052]), the recalculation and updating of the transport-layer headers that contain pseudo-headers need to be performed. Translators MUST do this for TCP and ICMP packets and for UDP packets that contain a UDP checksum (i.e., the UDP checksum field is not zero).

For UDP packets that do not contain a UDP checksum (i.e., the UDP checksum field is zero), the translator SHOULD provide a configuration function to allow:

1. Dropping the packet and generating a system management event that specifies at least the IP addresses and port numbers of the packet.

2. Calculating an IPv6 checksum and forwarding the packet (which has performance implications).

   A stateless translator cannot compute the UDP checksum of fragmented packets, so when a stateless translator receives the first fragment of a fragmented UDP IPv4 packet and the checksum field is zero, the translator SHOULD drop the packet and generate a system management event that specifies at least the IP addresses and port numbers in the packet.

   For a stateful translator, the handling of fragmented UDP IPv4 packets with a zero checksum is discussed in [RFC6146]), Section 3.1.

Other transport protocols (e.g., DCCP) are OPTIONAL to support. In order to ease debugging and troubleshooting, translators MUST forward all transport protocols as described in the "Next Header" step of Section 4.1.

4.6. Knowing When to Translate

If the IP/ICMP translator also provides a normal forwarding function, and the destination IPv4 address is reachable by a more specific route without translation, the translator MUST forward it without translating it. Otherwise, when an IP/ICMP translator receives an IPv4 datagram addressed to an IPv4 destination representing a host in the IPv6 domain, the packet MUST be translated to IPv6.

5. Translating from IPv6 to IPv4

When an IP/ICMP translator receives an IPv6 datagram addressed to a destination towards the IPv4 domain, it translates the IPv6 header of the received IPv6 packet into an IPv4 header. The original IPv6 header on the packet is removed and replaced by an IPv4 header. Since the ICMPv6 [RFC4443], TCP [RFC0793], UDP [RFC0768], and DCCP [RFC4340] headers contain checksums that cover the IP header, if the address mapping algorithm is not checksum neutral, the checksum MUST be evaluated before translation and the ICMP and transport-layer headers MUST be updated. The data portion of the packet is left unchanged. The IP/ICMP translator then forwards the packet based on the IPv4 destination address.

```
           +-------------+                 +-------------+
           |    IPv6     |                 |    IPv4     |
           |   Header    |                 |   Header    |
           +-------------+                 +-------------+
           |  Fragment   |                 |  Transport  |
           |   Header    |      ===>       |   Layer     |
           |(if present) |                 |   Header    |
           +-------------+                 +-------------+
           |  Transport  |                 |             |
           |   Layer     |                 ~    Data     ~
           |   Header    |                 |             |
           +-------------+                 +-------------+
           |             |
           ~    Data     ~
           |             |
           +-------------+
```

Figure 5: IPv6-to-IPv4 Translation

There are some differences between IPv6 and IPv4 (in the areas of fragmentation and the minimum link MTU) that affect the translation. An IPv6 link has to have an MTU of 1280 bytes or greater. The corresponding limit for IPv4 is 68 bytes. Path MTU discovery across a translator relies on ICMP Packet Too Big messages being received and processed by IPv6 hosts, including an ICMP Packet Too Big that indicates the MTU is less than the IPv6 minimum MTU. This requirement is described in Section 5 of [RFC2460] (for IPv6's 1280-octet minimum MTU) and Section 5 of [RFC1883] (for IPv6's previous 576-octet minimum MTU).

In an environment where an ICMPv4 Packet Too Big message is translated to an ICMPv6 Packet Too Big message, and the ICMPv6 Packet Too Big message is successfully delivered to and correctly processed by the IPv6 hosts (e.g., a network owned/operated by the same entity that owns/operates the translator), the translator can rely on IPv6 hosts sending subsequent packets to the same IPv6 destination with IPv6 Fragment Headers. In such an environment, when the translator receives an IPv6 packet with a Fragment Header, the translator SHOULD generate the IPv4 packet with a cleared Don't Fragment bit, and with its identification value from the IPv6 Fragment Header, for all of the IPv6 fragments (MF=0 or MF=1).

In an environment where an ICMPv4 Packet Too Big message is filtered (by a network firewall or by the host itself) or not correctly processed by the IPv6 hosts, the IPv6 host will never generate an IPv6 packet with the IPv6 Fragment Header. In such an environment, the translator SHOULD set the IPv4 Don't Fragment bit. While setting the Don't Fragment bit may create PMTUD black holes [RFC2923] if there are IPv4 links smaller than 1260 octets, this is considered safer than causing IPv4 reassembly errors [RFC4963].

Other than the special rules for handling fragments and path MTU discovery, the actual translation of the packet header consists of a simple translation as defined below. Note that ICMPv6 packets require special handling in order to translate the contents of ICMPv6 error messages and also to remove the ICMPv6 pseudo-header checksum.

The translator SHOULD make sure that the packets belonging to the same flow leave the translator in the same order in which they arrived.

5.1. Translating IPv6 Headers into IPv4 Headers

If there is no IPv6 Fragment Header, the IPv4 header fields are set as follows:

Version: 4

Internet Header Length: 5 (no IPv4 options)

Type of Service (TOS) Octet: By default, copied from the IPv6 Traffic Class (all 8 bits). According to [RFC2474], the semantics of the bits are identical in IPv4 and IPv6. However, in some IPv4 environments, these bits might be used with the old semantics of "Type Of Service and Precedence". An implementation of a translator SHOULD provide the ability to ignore the IPv6 traffic class and always set the IPv4 TOS Octet to a specified value. In addition, if the translator is at an administrative boundary, the filtering and update considerations of [RFC2475] may be applicable.

Total Length: Payload length value from the IPv6 header, plus the size of the IPv4 header.

Identification: All zero. In order to avoid black holes caused by ICMPv4 filtering or non-[RFC2460]-compatible IPv6 hosts (a workaround is discussed in Section 6), the translator MAY provide a function to generate the identification value if the packet size is greater than 88 bytes and less than or equal to 1280 bytes.

The translator SHOULD provide a method for operators to enable or disable this function.

Flags: The More Fragments flag is set to zero. The Don't Fragment (DF) flag is set to one. In order to avoid black holes caused by ICMPv4 filtering or non-[RFC2460]-compatible IPv6 hosts (a workaround is discussed in Section 6), the translator MAY provide a function as follows. If the packet size is greater than 88 bytes and less than or equal to 1280 bytes, it sets the DF flag to zero; otherwise, it sets the DF flag to one. The translator SHOULD provide a method for operators to enable or disable this function.

Fragment Offset: All zeros.

Time to Live: Time to Live is derived from Hop Limit value in IPv6 header. Since the translator is a router, as part of forwarding the packet it needs to decrement either the IPv6 Hop Limit (before the translation) or the IPv4 TTL (after the translation). As part of decrementing the TTL or Hop Limit the translator (as any router) MUST check for zero and send the ICMPv4 "TTL Exceeded" or ICMPv6 "Hop Limit Exceeded" error.

Protocol: The IPv6-Frag (44) header is handled as discussed in Section 5.1.1. ICMPv6 (58) is changed to ICMPv4 (1), and the payload is translated as discussed in Section 5.2. The IPv6 headers HOPOPT (0), IPv6-Route (43), and IPv6-Opts (60) are skipped over during processing as they have no meaning in IPv4. For the first 'next header' that does not match one of the cases above, its Next Header value (which contains the transport protocol number) is copied to the protocol field in the IPv4 header. This means that all transport protocols are translated.

Note: Some translated protocols will fail at the receiver for various reasons: some are known to fail when translated (e.g., IPsec Authentication Header (51)), and others will fail checksum validation if the address translation is not checksum neutral [RFC6052] and the translator does not update the transport protocol's checksum (because the translator doesn't support recalculating the checksum for that transport protocol; see Section 5.5).

Header Checksum: Computed once the IPv4 header has been created.

Source Address: In the stateless mode (which is to say that if the IPv6 source address is within the range of a configured IPv6 translation prefix), the IPv4 source address is derived from the IPv6 source address per [RFC6052], Section 2.3. Note that the original IPv6 source address is an IPv4-translatable address. A workflow example of stateless translation is shown in Appendix A of this document. If the translator only supports stateless mode and if the IPv6 source address is not within the range of configured IPv6 prefix(es), the translator SHOULD drop the packet and respond with an ICMPv6 "Destination Unreachable, Source address failed ingress/egress policy" (Type 1, Code 5).

In the stateful mode, which is to say that if the IPv6 source address is not within the range of any configured IPv6 stateless translation prefix, the IPv4 source address and transport-layer source port corresponding to the IPv4-related IPv6 source address and source port are derived from the Binding Information Bases (BIBs) as described in [RFC6146].

In stateless and stateful modes, if the translator gets an illegal source address (e.g., ::1, etc.), the translator SHOULD silently drop the packet.

Destination Address: The IPv4 destination address is derived from the IPv6 destination address of the datagram being translated per [RFC6052], Section 2.3. Note that the original IPv6 destination address is an IPv4-converted address.

If a Routing header with a non-zero Segments Left field is present, then the packet MUST NOT be translated, and an ICMPv6 "parameter problem/erroneous header field encountered" (Type 4, Code 0) error message, with the Pointer field indicating the first byte of the Segments Left field, SHOULD be returned to the sender.

5.1.1. IPv6 Fragment Processing

If the IPv6 packet contains a Fragment Header, the header fields are set as above with the following exceptions:

Total Length: Payload length value from IPv6 header, minus 8 for the Fragment Header, plus the size of the IPv4 header.

Identification: Copied from the low-order 16 bits in the Identification field in the Fragment Header.

Flags: The IPv4 More Fragments (MF) flag is copied from the M flag in the IPv6 Fragment Header. The IPv4 Don't Fragment (DF) flag is cleared (set to zero), allowing this packet to be further fragmented by IPv4 routers.

Fragment Offset: Copied from the Fragment Offset field of the IPv6 Fragment Header.

Protocol: For ICMPv6 (58), it is changed to ICMPv4 (1); otherwise, extension headers are skipped, and the Next Header field is copied from the last IPv6 header.

If a translated packet with DF set to 1 will be larger than the MTU of the next-hop interface, then the translator MUST drop the packet and send the ICMPv6 Packet Too Big (Type 2, Code 0) error message to the IPv6 host with an adjusted MTU in the ICMPv6 message.

5.2. Translating ICMPv6 Headers into ICMPv4 Headers

If a non-checksum-neutral translation address is being used, ICMPv6 messages MUST have their ICMPv4 checksum field be updated as part of the translation since ICMPv6 (unlike ICMPv4) includes a pseudo-header in the checksum just like UDP and TCP.

In addition, all ICMP packets MUST have the Type translated and, for ICMP error messages, the included IP header also MUST be translated. Note that the IPv6 addresses in the IPv6 header may not be IPv4-translatable addresses and there will be no corresponding IPv4 addresses representing this IPv6 address. In this case, the translator can do stateful translation. A mechanism by which the translator can instead do stateless translation of this address is left for future work.

The actions needed to translate various ICMPv6 messages are:

ICMPv6 informational messages:

Echo Request and Echo Reply (Type 128 and 129): Adjust the Type values to 8 and 0, respectively, and adjust the ICMP checksum both to take the type change into account and to exclude the ICMPv6 pseudo-header.

MLD Multicast Listener Query/Report/Done (Type 130, 131, 132): Single-hop message. Silently drop.

Neighbor Discover messages (Type 133 through 137): Single-hop message. Silently drop.

Unknown informational messages: Silently drop.

ICMPv6 error messages:

Destination Unreachable (Type 1) Set the Type to 3, and adjust the ICMP checksum both to take the type/code change into account and to exclude the ICMPv6 pseudo-header.

Translate the Code as follows:

Code 0 (No route to destination): Set the Code to 1 (Host unreachable).

Code 1 (Communication with destination administratively prohibited): Set the Code to 10 (Communication with destination host administratively prohibited).

Code 2 (Beyond scope of source address): Set the Code to 1 (Host unreachable). Note that this error is very unlikely since an IPv4-translatable source address is typically considered to have global scope.

Code 3 (Address unreachable): Set the Code to 1 (Host unreachable).

Code 4 (Port unreachable): Set the Code to 3 (Port unreachable).

Other Code values: Silently drop.

Packet Too Big (Type 2): Translate to an ICMPv4 Destination Unreachable (Type 3) with Code 4, and adjust the ICMPv4 checksum both to take the type change into account and to exclude the ICMPv6 pseudo-header. The MTU field MUST be adjusted for the difference between the IPv4 and IPv6 header sizes, taking into account whether or not the packet in error includes a Fragment Header, i.e., minimum(advertised MTU-20, MTU_of_IPv4_nexthop, (MTU_of_IPv6_nexthop)-20).

See also the requirements in Section 6.

Time Exceeded (Type 3): Set the Type to 11, and adjust the ICMPv4 checksum both to take the type change into account and to exclude the ICMPv6 pseudo-header. The Code is unchanged.

Parameter Problem (Type 4): Translate the Type and Code as follows, and adjust the ICMPv4 checksum both to take the type/code change into account and to exclude the ICMPv6 pseudo-header.

Translate the Code as follows:

Code 0 (Erroneous header field encountered): Set to Type 12, Code 0, and update the pointer as defined in Figure 6. (If the Original IPv6 Pointer Value is not listed or the Translated IPv4 Pointer Value is listed as "n/a", silently drop the packet.)

Code 1 (Unrecognized Next Header type encountered): Translate this to an ICMPv4 protocol unreachable (Type 3, Code 2).

Code 2 (Unrecognized IPv6 option encountered): Silently drop.

Unknown error messages: Silently drop.

Original IPv6 Pointer Value		Translated IPv4 Pointer Value	
0	Version/Traffic Class	0	Version/IHL, Type Of Ser
1	Traffic Class/Flow Label	1	Type Of Service
2,3	Flow Label	n/a	
4,5	Payload Length	2	Total Length
6	Next Header	9	Protocol
7	Hop Limit	8	Time to Live
8-23	Source Address	12	Source Address
24-39	Destination Address	16	Destination Address

Figure 6: Pointer Value for Translating from IPv6 to IPv4

ICMP Error Payload: If the received ICMPv6 packet contains an ICMPv6 Extension [RFC4884], the translation of the ICMPv6 packet will cause the ICMPv4 packet to change length. When this occurs, the ICMPv6 Extension length attribute MUST be adjusted accordingly (e.g., shorter due to the translation from IPv6 to IPv4). For extensions not defined in [RFC4884], the translator passes the extensions as opaque bit strings and any IPv6 address literals contained therein will not be translated to IPv4 address literals; this may cause problems with processing of those ICMP extensions.

5.3. Translating ICMPv6 Error Messages into ICMPv4

There are some differences between the ICMPv4 and the ICMPv6 error message formats as detailed above. The ICMP error messages containing the packet in error MUST be translated just like a normal IP packet. The translation of this "packet in error" is likely to change the length of the datagram; thus, the Total Length field in the outer IPv4 header MUST be updated.

```
            +-------------+                 +-------------+
            |    IPv6     |                 |    IPv4     |
            |   Header    |                 |   Header    |
            +-------------+                 +-------------+
            |   ICMPv6    |                 |   ICMPv4    |
            |   Header    |                 |   Header    |
            +-------------+                 +-------------+
            |    IPv6     |      ===>       |    IPv4     |
            |   Header    |                 |   Header    |
            +-------------+                 +-------------+
            |   Partial   |                 |   Partial   |
            |  Transport- |                 |  Transport- |
            |   Layer     |                 |   Layer     |
            |   Header    |                 |   Header    |
            +-------------+                 +-------------+
```

Figure 7: IPv6-to-IPv4 ICMP Error Translation

The translation of the inner IP header can be done by invoking the function that translated the outer IP headers. This process MUST stop at the first embedded header and drop the packet if it contains more embedded headers. Note that the IPv6 addresses in the IPv6 header may not be IPv4-translatable addresses, and there will be no corresponding IPv4 addresses. In this case, the translator can do stateful translation. A mechanism by which the translator can instead do stateless translation is left for future work.

5.4. Generation of ICMPv6 Error Messages

If the IPv6 packet is discarded, then the translator SHOULD send back an ICMPv6 error message to the original sender of the packet, unless the discarded packet is itself an ICMPv6 message.

If the ICMPv6 error message is being sent because the IPv6 source address is not an IPv4-translatable address and the translator is stateless, the ICMPv6 message (if sent) MUST have Type 1 and Code 5 (Source address failed ingress/egress policy). In other cases, the ICMPv6 message MUST have Type 1 (Destination Unreachable) and Code 1 (Communication with destination administratively prohibited), unless otherwise specified in this document or [RFC6146]. The translator SHOULD allow an administrator to configure whether the ICMPv6 error messages are sent, rate-limited, or not sent.

5.5. Transport-Layer Header Translation

If the address translation algorithm is not checksum neutral (see Section 4.1 of [RFC6052]), the recalculation and updating of the transport-layer headers that contain pseudo-headers need to be performed. Translators MUST do this for TCP, UDP, and ICMP.

Other transport protocols (e.g., DCCP) are OPTIONAL to support. In order to ease debugging and troubleshooting, translators MUST forward all transport protocols as described in the "Protocol" step of Section 5.1.

5.6. Knowing When to Translate

If the IP/ICMP translator also provides a normal forwarding function, and the destination address is reachable by a more specific route without translation, the router MUST forward it without translating it. When an IP/ICMP translator receives an IPv6 datagram addressed to an IPv6 address representing a host in the IPv4 domain, the IPv6 packet MUST be translated to IPv4.

6. Special Considerations for ICMPv6 Packet Too Big

Two recent studies analyzed the behavior of IPv6-capable web servers on the Internet and found that approximately 95% responded as expected to an IPv6 Packet Too Big that indicated MTU = 1280, but only 43% responded as expected to an IPv6 Packet Too Big that indicated an MTU < 1280. It is believed that firewalls violating Section 4.3.1 of [RFC4890] are at fault. Both failures (the 5% wrong response when MTU = 1280 and the 57% wrong response when MTU < 1280) will cause PMTUD black holes [RFC2923]. Unfortunately, the translator cannot improve the failure rate of the first case (MTU = 1280), but the translator can improve the failure rate of the second case (MTU < 1280). There are two approaches to resolving the problem with sending ICMPv6 messages indicating an MTU < 1280. It SHOULD be possible to configure a translator for either of the two approaches.

The first approach is to constrain the deployment of the IPv6/IPv4 translator by observing that four of the scenarios intended for stateless IPv6/IPv4 translators do not have IPv6 hosts on the Internet (Scenarios 1, 2, 5, and 6 described in [RFC6144], which refer to "An IPv6 network"). In these scenarios, IPv6 hosts, IPv6-host-based firewalls, and IPv6 network firewalls can be administered in compliance with Section 4.3.1 of [RFC4890] and therefore avoid the problem witnessed with IPv6 hosts on the Internet.

The second approach is necessary if the translator has IPv6 hosts, IPv6-host-based firewalls, or IPv6 network firewalls that do not (or cannot) comply with Section 5 of [RFC2460] -- such as IPv6 hosts on the Internet. This approach requires the translator to do the following:

1. In the IPv4-to-IPv6 direction: if the MTU value of ICMPv4 Packet Too Big (PTB) messages is less than 1280, change it to 1280. This is intended to cause the IPv6 host and IPv6 firewall to process the ICMP PTB message and generate subsequent packets to this destination with an IPv6 Fragment Header.

   Note: Based on recent studies, this is effective for 95% of IPv6 hosts on the Internet.

2. In the IPv6-to-IPv4 direction:

   A. If there is a Fragment Header in the IPv6 packet, the last 16 bits of its value MUST be used for the IPv4 identification value.

   B. If there is no Fragment Header in the IPv6 packet:

      a. If the packet is less than or equal to 1280 bytes:

         - The translator SHOULD set DF to 0 and generate an IPv4 identification value.

         - To avoid the problems described in [RFC4963], it is RECOMMENDED that the translator maintain 3-tuple state for generating the IPv4 identification value.

      b. If the packet is greater than 1280 bytes, the translator SHOULD set the IPv4 DF bit to 1.

7. Security Considerations

The use of stateless IP/ICMP translators does not introduce any new security issues beyond the security issues that are already present in the IPv4 and IPv6 protocols and in the routing protocols that are used to make the packets reach the translator.

There are potential issues that might arise by deriving an IPv4 address from an IPv6 address -- particularly addresses like broadcast or loopback addresses and the non-IPv4-translatable IPv6 addresses, etc. [RFC6052] addresses these issues.

As with network address translation of IPv4 to IPv4, the IPsec Authentication Header [RFC4302] cannot be used across an IPv6-to-IPv4 translator.

As with network address translation of IPv4 to IPv4, packets with tunnel mode Encapsulating Security Payload (ESP) can be translated since tunnel mode ESP does not depend on header fields prior to the ESP header. Similarly, transport mode ESP will fail with IPv6-to-IPv4 translation unless checksum-neutral addresses are used. In both cases, the IPsec ESP endpoints will normally detect the presence of the translator and encapsulate ESP in UDP packets [RFC3948].

8. Acknowledgements

This is under development by a large group of people. Those who have posted to the list during the discussion include Alexey Melnikov, Andrew Sullivan, Andrew Yourtchenko, Brian Carpenter, Dan Wing, Dave Thaler, David Harrington, Ed Jankiewicz, Hiroshi Miyata, Iljitsch van Beijnum, Jari Arkko, Jerry Huang, John Schnizlein, Jouni Korhonen, Kentaro Ebisawa, Kevin Yin, Magnus Westerlund, Marcelo Bagnulo Braun, Margaret Wasserman, Masahito Endo, Phil Roberts, Philip Matthews, Reinaldo Penno, Remi Denis-Courmont, Remi Despres, Sean Turner, Senthil Sivakumar, Simon Perreault, Stewart Bryant, Tim Polk, Tero Kivinen, and Zen Cao.

9. References

9.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC1883] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 1883, December 1995.

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

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

[RFC2765] Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, January 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

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

[RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, April 2007.

[RFC5382] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008.

[RFC5771] Cotton, M., Vegoda, L., and D. Meyer, "IANA Guidelines for IPv4 Multicast Address Assignments", BCP 51, RFC 5771, March 2010.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

[RFC6146] Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

9.2. Informative References

[RFC0879] Postel, J., "TCP maximum segment size and related topics", RFC 879, November 1983.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, October 1999.

[RFC2766] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, September 2000.

[RFC3307] Haberman, B., "Allocation Guidelines for IPv6 Multicast Addresses", RFC 3307, August 2002.

[RFC3590] Haberman, B., "Source Address Selection for the Multicast Listener Discovery (MLD) Protocol", RFC 3590, September 2003.

[RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.

[RFC3849] Huston, G., Lord, A., and P. Smith, "IPv6 Address Prefix Reserved for Documentation", RFC 3849, July 2004.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4890] Davies, E. and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls", RFC 4890, May 2007.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.

[RFC4966] Aoun, C. and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", RFC 4966, July 2007.

[RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", RFC 5737, January 2010.

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, April 2011.

Appendix A. Stateless Translation Workflow Example

A stateless translation workflow example is depicted in the following figure. The documentation address blocks 2001:db8::/32 [RFC3849], 192.0.2.0/24, and 198.51.100.0/24 [RFC5737] are used in this example.

```
+--------------+                   +--------------+
| IPv4 network |                   | IPv6 network |
|              |     +-------+     |              |
|   +----+     |-----| XLAT  |---- |  +----+      |
|   | H4 |-----|     +-------+     |--| H6 |      |
|   +----+     |                   |  +----+      |
+--------------+                   +--------------+
```

Figure 8

A translator (XLAT) connects the IPv6 network to the IPv4 network. This XLAT uses the Network-Specific Prefix (NSP) 2001:db8:100::/40 defined in [RFC6052] to represent IPv4 addresses in the IPv6 address space (IPv4-converted addresses) and to represent IPv6 addresses (IPv4-translatable addresses) in the IPv4 address space. In this example, 192.0.2.0/24 is the IPv4 block of the corresponding IPv4-translatable addresses.

Based on the address mapping rule, the IPv6 node H6 has an IPv4-translatable IPv6 address 2001:db8:1c0:2:21:: (address mapping from 192.0.2.33). The IPv4 node H4 has IPv4 address 198.51.100.2.

The IPv6 routing is configured in such a way that the IPv6 packets addressed to a destination address in 2001:db8:100::/40 are routed to the IPv6 interface of the XLAT.

The IPv4 routing is configured in such a way that the IPv4 packets addressed to a destination address in 192.0.2.0/24 are routed to the IPv4 interface of the XLAT.

A.1. H6 Establishes Communication with H4

The steps by which H6 establishes communication with H4 are:

1. H6 performs the destination address mapping, so the IPv4-converted address 2001:db8:1c6:3364:2:: is formed from 198.51.100.2 based on the address mapping algorithm [RFC6052].

2. H6 sends a packet to H4. The packet is sent from a source address 2001:db8:1c0:2:21:: to a destination address 2001:db8:1c6:3364:2::.

3. The packet is routed to the IPv6 interface of the XLAT (since IPv6 routing is configured that way).

4. The XLAT receives the packet and performs the following actions:

   * The XLAT translates the IPv6 header into an IPv4 header using the IP/ICMP Translation Algorithm defined in this document.

   * The XLAT includes 192.0.2.33 as the source address in the packet and 198.51.100.2 as the destination address in the packet. Note that 192.0.2.33 and 198.51.100.2 are extracted directly from the source IPv6 address 2001:db8:1c0:2:21:: (IPv4-translatable address) and destination IPv6 address 2001:db8:1c6:3364:2:: (IPv4-converted address) of the received IPv6 packet that is being translated.

5. The XLAT sends the translated packet out of its IPv4 interface, and the packet arrives at H4.

6. H4 node responds by sending a packet with destination address 192.0.2.33 and source address 198.51.100.2.

7. The packet is routed to the IPv4 interface of the XLAT (since IPv4 routing is configured that way). The XLAT performs the following operations:

   * The XLAT translates the IPv4 header into an IPv6 header using the IP/ICMP Translation Algorithm defined in this document.

   * The XLAT includes 2001:db8:1c0:2:21:: as the destination address in the packet and 2001:db8:1c6:3364:2:: as the source address in the packet. Note that 2001:db8:1c0:2:21:: and 2001:db8:1c6:3364:2:: are formed directly from the destination IPv4 address 192.0.2.33 and the source IPv4 address 198.51.100.2 of the received IPv4 packet that is being translated.

8. The translated packet is sent out of the IPv6 interface to H6.

The packet exchange between H6 and H4 continues until the session is finished.

A.2. H4 Establishes Communication with H6

The steps by which H4 establishes communication with H6 are:

1. H4 performs the destination address mapping, so 192.0.2.33 is formed from the IPv4-translatable address 2001:db8:1c0:2:21:: based on the address mapping algorithm [RFC6052].

2. H4 sends a packet to H6. The packet is sent from a source address 198.51.100.2 to a destination address 192.0.2.33.

3. The packet is routed to the IPv4 interface of the XLAT (since IPv4 routing is configured that way).

4. The XLAT receives the packet and performs the following actions:

   * The XLAT translates the IPv4 header into an IPv6 header using the IP/ICMP Translation Algorithm defined in this document.

   * The XLAT includes 2001:db8:1c6:3364:2:: as the source address in the packet and 2001:db8:1c0:2:21:: as the destination address in the packet. Note that 2001:db8:1c6:3364:2:: (IPv4-converted address) and 2001:db8:1c0:2:21:: (IPv4-translatable address) are obtained directly from the source IPv4 address 198.51.100.2 and destination IPv4 address 192.0.2.33 of the received IPv4 packet that is being translated.

5. The XLAT sends the translated packet out its IPv6 interface, and the packet arrives at H6.

6. H6 node responds by sending a packet with destination address 2001:db8:1c6:3364:2:: and source address 2001:db8:1c0:2:21::.

7. The packet is routed to the IPv6 interface of the XLAT (since IPv6 routing is configured that way). The XLAT performs the following operations:

   * The XLAT translates the IPv6 header into an IPv4 header using the IP/ICMP Translation Algorithm defined in this document.

   * The XLAT includes 198.51.100.2 as the destination address in the packet and 192.0.2.33 as the source address in the packet. Note that 198.51.100.2 and 192.0.2.33 are formed directly from the destination IPv6 address 2001:db8:1c6:3364:2:: and source IPv6 address 2001:db8:1c0:2:21:: of the received IPv6 packet that is being translated.

8. The translated packet is sent out the IPv4 interface to H4.

The packet exchange between H4 and H6 continues until the session is finished.

Authors' Addresses

Xing Li
CERNET Center/Tsinghua University
Room 225, Main Building, Tsinghua University
Beijing, 100084
China

Phone: +86 10-62785983
EMail: xing@cernet.edu.cn

Congxiao Bao
CERNET Center/Tsinghua University
Room 225, Main Building, Tsinghua University
Beijing, 100084
China

Phone: +86 10-62785983
EMail: congxiao@cernet.edu.cn

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA

Phone: +1-408-526-4257
EMail: fred@cisco.com