                  IP/ICMP Translation Algorithm (rfc6145bis)
                        draft-bao-v6ops-rfc6145bis-06

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 6145.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction and Motivation

This document obsoletes [RFC6145].

Readers of this document are expected to have read and understood the
framework described in [RFC6144]. Implementations of this IPv4/IPv6
translation specification MUST support one or more address mapping
algorithms which are defined in Section 6.

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     |
          |               |   |             |
           \              |   |            /
            \\          //     \\         //
              --------           ---------
```
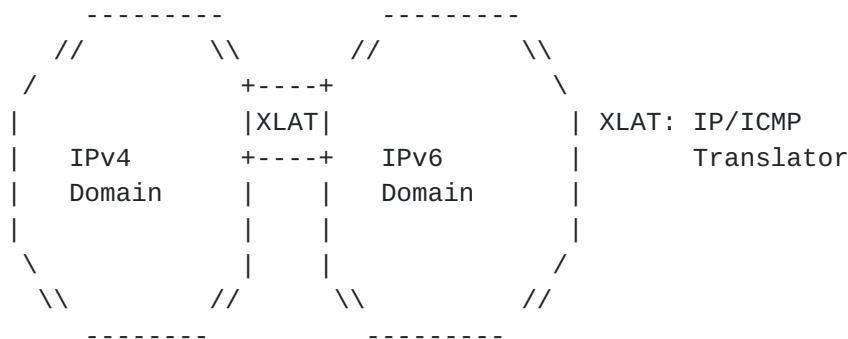
Figure 1: IPv4-IPv6 Translation Model

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 [RFC6145], 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 IP/ICMP
translators.

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]. An
example of experiments of the multicast address mapping can be found
in [RFC6219].

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, or they might have contiguous routing
connectivity but are interacting via masking addresses (i.e.
hairpinning) and hence are forced to have their communications
translated.

In the stateless mode, an IP/ICMP translator will convert IPv4
addresses to IPv6 and vice versa solely based on the configuration of
the stateless IP/ICMP translator and information contained within the
packet being translated. For example, for the default behavior
defined in [RFC6052], 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. Other
stateless translation algorithms are defined in Section 6. The
stateless translator does not keep any dynamic session or binding
state, thus there is no requirement that the packets in a single
session or flow traverses a single translator.

In the stateful mode, a specific IPv6 address range (consisting of
IPv4-converted IPv6 addresses) will typically represent IPv4 systems.
The IPv6 nodes may use any IPv6 addresses [RFC4291] except in that
range. A stateful IP/ICMP translator continuously maintains a
dynamic translation table containing bindings between the IPv4 and
IPv6 addresses, and likely also the Layer-4 identifiers, that are
used in the translated packets. The exact address translations of
any given packet thus become dependent on how packets belonging to
the same session or flow have been translated. For this reason,
stateful translation generally requires that all packets belonging to
a single flow must traverse the same translator.

In order to be able to successfully translate a packet from IPv4 to

IPv6 or vice versa, the translator must implement an address mapping
algorithm. This document does not specify any such algorithms,
instead these are referenced from Section 6.

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 6145

The changes from RFC 6145 are the following:

1. Insert the notes for the IPv6 extension header handling
    [Erratum].

2. Deprecate the algorithm which generates the IPv6 atomic
    fragments, as a result of the analysis in
    [I-D.ietf-6man-deprecate-atomfrag-generation] and specification
    in [I-D.ietf-6man-rfc2460bis].

3. Insert the notes for stateless source address mapping for ICMPv6
    packets [RFC6791].

4. Support new address mapping algorithms and move the discussion of
    these algorithms to Section 6.

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    ~
                                            |             |
                                            +-------------+
```
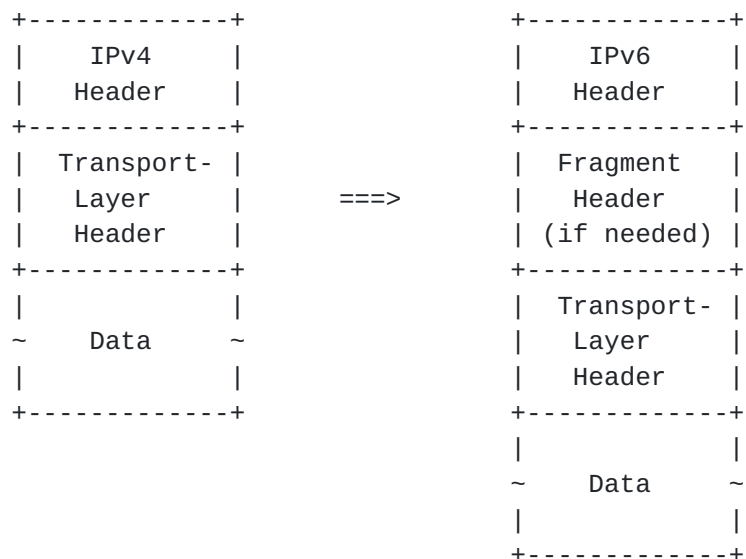
                   Figure 2: IPv4-to-IPv6 Translation

   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 MUST 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 MUST NOT
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.

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 a user-defined length (hereinafter referred to as
"lowest-ipv6-mtu", and which defaults to 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 lowest-
ipv6-mtu, 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 MUST 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.

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: Mapped to an IPv6 address based on the algorithms
   presented in Section 6.

   If the translator gets an illegal source address (e.g., 0.0.0.0,
   127.0.0.1, etc.), the translator SHOULD silently discard the
   packet (as discussed in Section 5.3.7 of [RFC1812]). Note when
   translating ICMPv4 Error Messages into ICMPv6, the "illegal"
   source address will be translated for the purpose of trouble
   shooting.

Destination Address: Mapped to an IPv6 address based on the
   algorithms presented in Section 6.

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 packet is a fragment
   or the DF bit is not set and the packet size is greater than the
   minimum IPv6 MTU in the network set by the translator configuration
   function), 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, but MUST NOT be
   set to a value smaller than the minimum IPv6 MTU (1280
   bytes). That is, it should be set to maximum(1280,
   minimum((MTU value in the Packet Too Big Message)+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,
   but that is larger than or equal to 1280.)

   See also the requirements in Section 7.

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 their included 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 (except the TTL value of the inner IPv4/IPv6 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   |
        +-------------+                   +-------------+
```
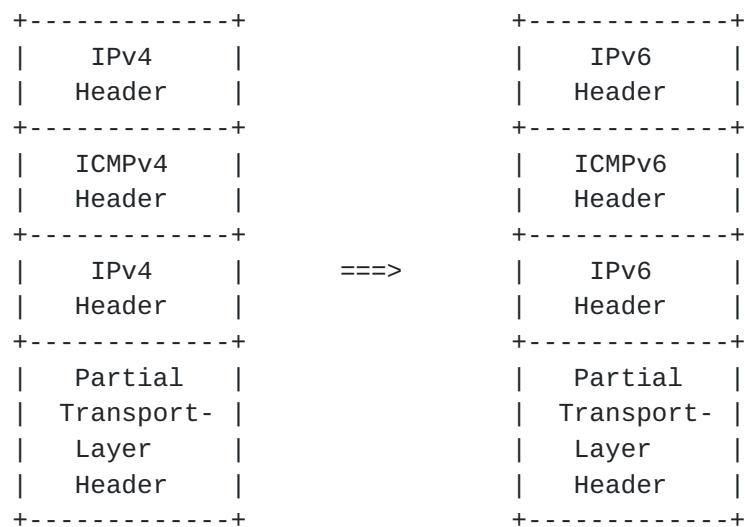
                Figure 4: IPv4-to-IPv6 ICMP Error Translation

   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 error
   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     ~
        |             |
        +-------------+
```
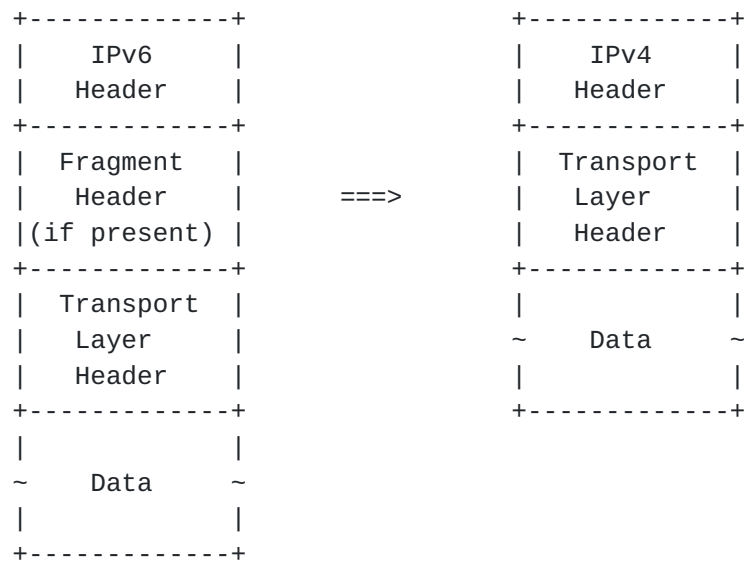
Figure 5: IPv6-to-IPv4 Translation

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.

The difference in the minimum MTUs of IPv4 and IPv6 is accommodated
as follows:

o When translating an ICMPv4 "Fragmentation Needed" packet, the
   indicated MTU in the resulting ICMPv6 "Packet Too Big" will never
   be set to a value lower than 1280. This ensures that the IPv6
   nodes will never have to encounter or handle Path MTU values lower
   than the minimum IPv6 link MTU of 1280. See Section 4.2.

o When the resulting IPv4 packet is smaller than or equal to 1260
   bytes, the translator MUST send the packet with a cleared Don't
   Fragment bit. Otherwise, the packet MUST be sent with the Don't
   Fragment bit set. See Section 5.1.

This approach allows Path MTU Discovery to operate end-to-end for
paths whose MTU are not smaller than minimum IPv6 MTU of 1280 (which
corresponds to MTU of 1260 in the IPv4 domain). On paths that have
IPv4 links with MTU < 1260, the IPv4 router(s) connected to those
links will fragment the packets in accordance with Section 2.3 of
[RFC0791].

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: Set according to a Fragment Identification generator
   at the translator.

Flags: The More Fragments flag is set to zero. The Don't Fragment
   (DF) flag is set as follows: If the size of the translated IPv4
   packet is less than or equal to 1260 bytes, it is set to zero;
   otherwise, it is set to one.

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: Mapped to an IPv4 address based on the algorithms
   presented in Section 6.

   If the translator gets an illegal source address (e.g., ::1,
   etc.), the translator SHOULD silently drop the packet.

Destination Address: Mapped to an IPv4 address based on the
   algorithms presented in Section 6.

If any of an IPv6 Hop-by-Hop Options header, Destination Options
header, or Routing header with the Segments Left field equal to zero
are present in the IPv6 packet, those IPv6 extension headers MUST be
ignored (i.e., there is no attempt to translate the extension
headers) and the packet translated normally. However, the Total
Length field and the Protocol field are adjusted to "skip" these
extension headers.

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: If the Next Header field of the Fragment Header is not
   an extension header (except ESP) then Total Length MUST be set to
   Payload Length value from IPv6 header, minus length of extension
   headers up to Fragmentation Header, minus 8 for the Fragment
   Header, plus the size of the IPv4 header. If the Next Header
   field of the Fragment Header is an extension header (except ESP)
   then the packet SHOULD be dropped and logged.

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: If the Next Header field of the Fragment Header is
   not an extension header (except ESP) then Fragment Offset MUST be
   copied from the Fragment Offset field of the IPv6 Fragment Header.
   If the Next Header field of the Fragment Header is an extension
   header (except ESP) then the packet SHOULD be dropped and logged.

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 an IPv6 packet that is smaller than or equal to 1280 bytes results
(after translation) in an IPv4 packet that is larger than the MTU of
the next-hop interface, then the translator MUST perform IPv4
fragmentation on that packet such that it can be transferred over the
constricting link.

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.

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((MTU value in the
Packet Too Big Message)-20, MTU_of_IPv4_nexthop,
(MTU_of_IPv6_nexthop)-20).

See also the requirements in Section 7.

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 (except the TTL value of the inner IPv4/IPv6 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    |
            +-------------+                 +-------------+
```
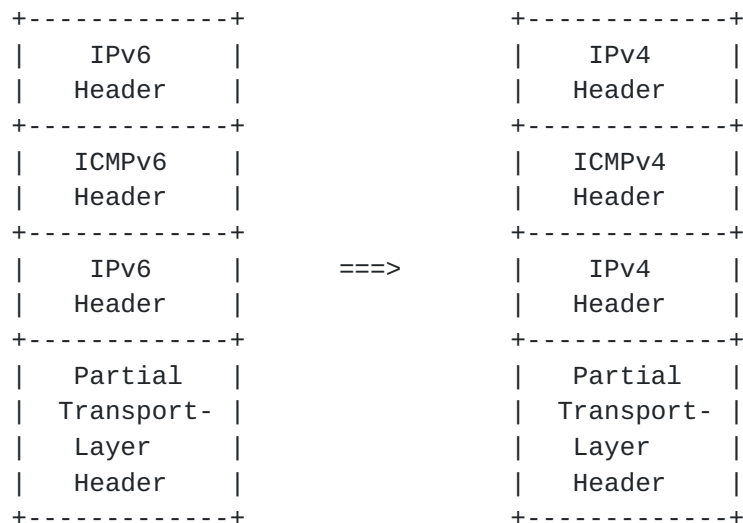
Figure 7: IPv6-to-IPv4 ICMP Error Translation

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.

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.

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. Mapping of IP Addresses

The translator MUST support stateless address mapping algorithm
defined in [RFC6052], which is the default behavior. A workflow
example is shown in Appendix A of this document. Note that [RFC7136]
updates [RFC4291] which allows the use of unicast addresses without
u-bit, as long as they're not derived from an IEEE MAC-layer address.
Therefore the address mapping algorithm defined in [RFC6219] also
complies with the IPv6 address architecture.

The stateless translator SHOULD support explicit address mapping
algorithm defined in [RFC7757].

The stateless translator SHOULD support [RFC6791] for handling ICMP/
ICMPv6 packets.

Implementations may support both stateless and stateful translation
modes (e.g., NAT64 [RFC6146]).

Implementations may support stateless NAT64 function, e.g., MAP-T CE or MAP-T BR [RFC7599].

7. Special Considerations for ICMPv6 Packet Too Big

A number of studies [I-D.ietf-6man-deprecate-atomfrag-generation] indicate that it not unusual for networks to drop ICMPv6 Packet Too Big error messages. Such packet drops will result in PMTUD blackholes [RFC2923], which can only be overcome with PLPMTUD [RFC4821].

8. IANA Considerations

This document makes no requests of the IANA.

9. 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].

10. Acknowledgements

The authors would like to thank Gandhar Gokhale, Wesley Eddy and Fernando Gont for the errata report of [RFC6145]. Thanks to Fernando Gont, Will(Shucheng) Liu and Tore Anderson for the security analysis and the suggestions of the updates concerning atomic fragments. Thanks to Tore Anderson and Alberto Leiva for the proposal of the explicit address mapping (EAM) algorithm.

11. References

11.1. Normative References

[RFC0768]   Postel, J., "User
                                   Datagram Protocol",
                                   STD 6, RFC 768,
                                   DOI 10.17487/RFC0768,
                                   August 1980, <http://
                                   www.rfc-editor.org/
                                   info/rfc768>.

[RFC0791]   Postel, J., "Internet
                                   Protocol", STD 5,
                                   RFC 791, DOI 10.17487/
                                   RFC0791,
                                   September 1981, <http:
                                   //www.rfc-editor.org/
                                   info/rfc791>.

[RFC0793]   Postel, J.,
                                   "Transmission Control
                                   Protocol", STD 7,
                                   RFC 793, DOI 10.17487/
                                   RFC0793,
                                   September 1981, <http:
                                   //www.rfc-editor.org/
                                   info/rfc793>.

[RFC1812]   Baker, F., Ed.,
                                   "Requirements for IP
                                   Version 4 Routers",
                                   RFC 1812,
                                   DOI 10.17487/RFC1812,
                                   June 1995, <http://
                                   www.rfc-editor.org/
                                   info/rfc1812>.

[RFC2119]   Bradner, S., "Key
                                   words for use in RFCs
                                   to Indicate
                                   Requirement Levels",
                                   BCP 14, RFC 2119,
                                   DOI 10.17487/RFC2119,
                                   March 1997, <http://
                                   www.rfc-editor.org/
                                   info/rfc2119>.

   [RFC3948]   Huttunen, A., Swander,
                                      B., Volpe, V.,
                                      DiBurro, L., and M.
                                      Stenberg, "UDP
                                      Encapsulation of IPsec
                                      ESP Packets",
                                      RFC 3948,
                                      DOI 10.17487/RFC3948,
                                      January 2005, <http://
                                      www.rfc-editor.org/
                                      info/rfc3948>.

   [RFC4291]   Hinden, R. and S.
                                      Deering, "IP Version 6
                                      Addressing
                                      Architecture",
                                      RFC 4291,
                                      DOI 10.17487/RFC4291,
                                      February 2006, <http:/
                                      /www.rfc-editor.org/
                                      info/rfc4291>.

   [RFC4340]   Kohler, E., Handley,
                                      M., and S. Floyd,
                                      "Datagram Congestion
                                      Control Protocol
                                      (DCCP)", RFC 4340,
                                      DOI 10.17487/RFC4340,
                                      March 2006, <http://
                                      www.rfc-editor.org/
                                      info/rfc4340>.

   [RFC4443]   Conta, A., Deering,
                                      S., and M. Gupta, Ed.,
                                      "Internet Control
                                      Message Protocol
                                      (ICMPv6) for the
                                      Internet Protocol
                                      Version 6 (IPv6)
                                      Specification",
                                      RFC 4443,
                                      DOI 10.17487/RFC4443,
                                      March 2006, <http://
                                      www.rfc-editor.org/
                                      info/rfc4443>.

   [RFC4884]   Bonica, R., Gan, D.,
                                      Tappan, D., and C.

                                            Pignataro, "Extended
                                            ICMP to Support Multi-
                                            Part Messages",
                                            RFC 4884,
                                            DOI 10.17487/RFC4884,
                                            April 2007, <http://
                                            www.rfc-editor.org/
                                            info/rfc4884>.

   [RFC5382]   Guha, S., Ed., Biswas,
                                            K., Ford, B.,
                                            Sivakumar, S., and P.
                                            Srisuresh, "NAT
                                            Behavioral
                                            Requirements for TCP",
                                            BCP 142, RFC 5382,
                                            DOI 10.17487/RFC5382,
                                            October 2008, <http://
                                            www.rfc-editor.org/
                                            info/rfc5382>.

   [RFC5771]   Cotton, M., Vegoda,
                                            L., and D. Meyer,
                                            "IANA Guidelines for
                                            IPv4 Multicast Address
                                            Assignments", BCP 51,
                                            RFC 5771,
                                            DOI 10.17487/RFC5771,
                                            March 2010, <http://
                                            www.rfc-editor.org/
                                            info/rfc5771>.

   [RFC6052]   Bao, C., Huitema, C.,
                                            Bagnulo, M.,
                                            Boucadair, M., and X.
                                            Li, "IPv6 Addressing
                                            of IPv4/IPv6
                                            Translators",
                                            RFC 6052,
                                            DOI 10.17487/RFC6052,
                                            October 2010, <http://
                                            www.rfc-editor.org/
                                            info/rfc6052>.

   [RFC6145]   Li, X., Bao, C., and
                                            F. Baker, "IP/ICMP
                                            Translation
                                            Algorithm", RFC 6145,

                                  DOI 10.17487/RFC6145,
                                  April 2011, <http://
                                  www.rfc-editor.org/
                                  info/rfc6145>.

   [RFC6146]   Bagnulo, M., Matthews,
                                  P., and I. van
                                  Beijnum, "Stateful
                                  NAT64: Network Address
                                  and Protocol
                                  Translation from IPv6
                                  Clients to IPv4
                                  Servers", RFC 6146,
                                  DOI 10.17487/RFC6146,
                                  April 2011, <http://
                                  www.rfc-editor.org/
                                  info/rfc6146>.

   [RFC6791]   Li, X., Bao, C., Wing,
                                  D., Vaithianathan, R.,
                                  and G. Huston,
                                  "Stateless Source
                                  Address Mapping for
                                  ICMPv6 Packets",
                                  RFC 6791,
                                  DOI 10.17487/RFC6791,
                                  November 2012, <http:/
                                  /www.rfc-editor.org/
                                  info/rfc6791>.

   [RFC7757]   Anderson, T. and A.
                                  Leiva Popper,
                                  "Explicit Address
                                  Mappings for Stateless
                                  IP/ICMP Translation",
                                  RFC 7757,
                                  DOI 10.17487/RFC7757,
                                  February 2016, <http:/
                                  /www.rfc-editor.org/
                                  info/rfc7757>.

11.2. Informative References

   [Erratum]   "Erratum: http://
                                  www.rfc-editor.org/
                                  errata_search.php/doc/html/
rfc
                                  6145".

[I-D.ietf-6man-deprecate-atomfrag-generation]   Gont, F., LIU, S., and
                                                 T. Anderson,
                                                 "Generation of IPv6
                                                 Atomic Fragments
                                                 Considered Harmful", d
                                                 raft-ietf-6man-
                                                 deprecate-atomfrag-
                                                 generation-05 (work in
                                                 progress),
                                                 January 2016.

[I-D.ietf-6man-rfc2460bis]   Deering, S. and B.
                                                 Hinden, "Internet
                                                 Protocol, Version 6
                                                 (IPv6) Specification",
                                                 draft-ietf-6man-
                                                 rfc2460bis-03 (work in
                                                 progress),
                                                 January 2016.

[RFC0879]   Postel, J., "The TCP
                                                 Maximum Segment Size
                                                 and Related Topics",
                                                 RFC 879, DOI 10.17487/
                                                 RFC0879,
                                                 November 1983, <http:/
                                                 /www.rfc-editor.org/
                                                 info/rfc879>.

[RFC1191]   Mogul, J. and S.
                                                 Deering, "Path MTU
                                                 discovery", RFC 1191,
                                                 DOI 10.17487/RFC1191,
                                                 November 1990, <http:/
                                                 /www.rfc-editor.org/
                                                 info/rfc1191>.

[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,
                                                 DOI 10.17487/RFC2474,
                                                 December 1998, <http:/
                                                 /www.rfc-editor.org/

info/rfc2474>.

[RFC2475]   Blake, S., Black, D.,
                                         Carlson, M., Davies,
                                         E., Wang, Z., and W.
                                         Weiss, "An
                                         Architecture for
                                         Differentiated
                                         Services", RFC 2475,
                                         DOI 10.17487/RFC2475,
                                         December 1998, <http:/
                                         /www.rfc-editor.org/
                                         info/rfc2475>.

[RFC2710]   Deering, S., Fenner,
                                         W., and B. Haberman,
                                         "Multicast Listener
                                         Discovery (MLD) for
                                         IPv6", RFC 2710,
                                         DOI 10.17487/RFC2710,
                                         October 1999, <http://
                                         www.rfc-editor.org/
                                         info/rfc2710>.

[RFC2923]   Lahey, K., "TCP
                                         Problems with Path MTU
                                         Discovery", RFC 2923,
                                         DOI 10.17487/RFC2923,
                                         September 2000, <http:
                                         //www.rfc-editor.org/
                                         info/rfc2923>.

[RFC3307]   Haberman, B.,
                                         "Allocation Guidelines
                                         for IPv6 Multicast
                                         Addresses", RFC 3307,
                                         DOI 10.17487/RFC3307,
                                         August 2002, <http://
                                         www.rfc-editor.org/
                                         info/rfc3307>.

[RFC3590]   Haberman, B., "Source
                                         Address Selection for
                                         the Multicast Listener
                                         Discovery (MLD)
                                         Protocol", RFC 3590,
                                         DOI 10.17487/RFC3590,
                                         September 2003, <http:

                                              //www.rfc-editor.org/
                                              info/rfc3590>.

        [RFC3810]   Vida, R., Ed. and L.
                                              Costa, Ed., "Multicast
                                              Listener Discovery
                                              Version 2 (MLDv2) for
                                              IPv6", RFC 3810,
                                              DOI 10.17487/RFC3810,
                                              June 2004, <http://
                                              www.rfc-editor.org/
                                              info/rfc3810>.

        [RFC3849]   Huston, G., Lord, A.,
                                              and P. Smith, "IPv6
                                              Address Prefix
                                              Reserved for
                                              Documentation",
                                              RFC 3849,
                                              DOI 10.17487/RFC3849,
                                              July 2004, <http://
                                              www.rfc-editor.org/
                                              info/rfc3849>.

        [RFC4302]   Kent, S., "IP
                                              Authentication
                                              Header", RFC 4302,
                                              DOI 10.17487/RFC4302,
                                              December 2005, <http:/
                                              /www.rfc-editor.org/
                                              info/rfc4302>.

        [RFC4821]   Mathis, M. and J.
                                              Heffner,
                                              "Packetization Layer
                                              Path MTU Discovery",
                                              RFC 4821,
                                              DOI 10.17487/RFC4821,
                                              March 2007, <http://
                                              www.rfc-editor.org/
                                              info/rfc4821>.

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

                                        DOI 10.17487/RFC5737,
                                        January 2010, <http://
                                        www.rfc-editor.org/
                                        info/rfc5737>.

   [RFC6144]   Baker, F., Li, X.,
                                        Bao, C., and K. Yin,
                                        "Framework for IPv4/
                                        IPv6 Translation",
                                        RFC 6144,
                                        DOI 10.17487/RFC6144,
                                        April 2011, <http://
                                        www.rfc-editor.org/
                                        info/rfc6144>.

   [RFC6219]   Li, X., Bao, C., Chen,
                                        M., Zhang, H., and J.
                                        Wu, "The China
                                        Education and Research
                                        Network (CERNET) IVI
                                        Translation Design and
                                        Deployment for the
                                        IPv4/IPv6 Coexistence
                                        and Transition",
                                        RFC 6219,
                                        DOI 10.17487/RFC6219,
                                        May 2011, <http://
                                        www.rfc-editor.org/
                                        info/rfc6219>.

   [RFC7136]   Carpenter, B. and S.
                                        Jiang, "Significance
                                        of IPv6 Interface
                                        Identifiers",
                                        RFC 7136,
                                        DOI 10.17487/RFC7136,
                                        February 2014, <http:/
                                        /www.rfc-editor.org/
                                        info/rfc7136>.

   [RFC7599]   Li, X., Bao, C., Dec,
                                        W., Ed., Troan, O.,
                                        Matsushima, S., and T.
                                        Murakami, "Mapping of
                                        Address and Port using
                                        Translation (MAP-T)",
                                        RFC 7599,
                                        DOI 10.17487/RFC7599,

July 2015, <http://
www.rfc-editor.org/
info/rfc7599>.

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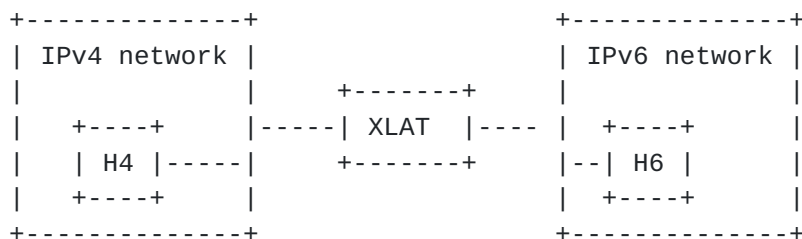
```
      +--------------+                  +--------------+
      | IPv4 network |                  | IPv6 network |
      |              |     +-------+     |              |
      |    +----+    |-----| XLAT  |---- |   +----+     |
      |    | H4 |-----|     +-------+    |--| H6 |      |
      | 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

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.

. **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

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

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

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

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

   Fred Baker
   Cisco Systems
   Santa Barbara, California 93117
   USA

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

   Tore Anderson
   Redpill Linpro
   Vitaminveien 1A
   0485 Oslo
   Norway

   Phone: +47 959 31 212
   EMail: tore@redpill-linpro.com
   URI: http://www.redpill-linpro.com

   Fernando Gont
   Huawei Technologies
   Evaristo Carriego 2644
   Haedo, Provincia de Buenos Aires 1706
   Argentina

   Phone: +54 11 4650 8472
   EMail: fgont@si6networks.com
   URI: http://www.si6networks.com