behave X. Li, Ed.
Internet-Draft C. Bao, Ed.
Intended status: Standards Track CERNET Center/Tsinghua University
Expires: August 25, 2009 F. Baker, Ed.
Cisco Systems
February 21, 2009

# IP/ICMP Translation Algorithm
# draft-baker-behave-v4v6-translation-02

## Abstract

This document specifies an update to the Stateless IP/ICMP Translation Algorithm (SIIT) described in RFC 2765. The algorithm translates between IPv4 and IPv6 packet headers (including ICMP headers).

This specification addresses both a stateless and a stateful mode. In the stateless mode, translation information is carried in the address itself, permitting both IPv4->IPv6 and IPv6->IPv4 session establishment with neither state nor configuration in the IP/ICMP translator. In the stateful mode, translation state is maintained between IPv4 address/transport_port tuples and IPv6 address/transport_port tuples, enabling IPv6 systems to open sessions with IPv4 systems. The choice of operational mode is made by the operator deploying the network and is critical to the operation of the applications using it.

Significant issues exist in the stateless and stateful modes that are not addressed in this document, related to the address assignment and the maintenance of the translation tables, respectively. This document confines itself to the actual translation.

## Acknowledgement of previous work

This document is a product of the 2008-2009 effort to define a replacement for NAT-PT. It is an update to and directly derivative from Erik Nordmark's [RFC2765], which similarly provides both stateless and stateful translation between IPv4 [RFC0791] and IPv6 [RFC2460], and between ICMPv4 [RFC0792] and ICMPv6 [RFC4443]. The original document was a product of the NGTRANS working group.

The changes in this document reflect five components:

1. Redescribing the network model to map to present and projected usage.

2. Moving the address format to the framework document, to coordinate with other drafts on the topic.

3. Description of both stateful and stateless operation.

4. Some changes in ICMP.

5. Updating references.

Requirements

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

Table of Contents

## 1. Introduction and Motivation

An understanding of the framework presented in [FRAMEWORK] is presumed in this document. With that remark...

The transition mechanisms specified in [RFC4213] handle the case of dual IPv4/IPv6 hosts interoperating with both dual hosts and IPv4-only hosts, which is needed early in the transition to IPv6. The dual hosts are assigned both an IPv4 and one or more IPv6 addresses. The number of available globally unique IPv4 addresses are becoming smaller and smaller as the Internet grows; we expect that there will be a desire to take advantage of the large IPv6 address and not require that every new Internet node have a permanently assigned IPv4 address.

The SIIT [RFC2765] is designed for the case for small networks (e.g., a single subnet) and for a site which has IPv6-only hosts in a dual IPv4/IPv6 network. This use assumes a mechanism for the IPv6 nodes to acquire a temporary address from the pool of IPv4 addresses. However, SIIT is not to be useful in the case when the IPv6 nodes to acquire temporary IPv4 addresses from a "distant" SIIT box operated by a different administration, or require that the IPv6 routing contain routes for IPv6-mapped addresses (The latter is known to be a very bad idea due to the size of the IPv4 routing table that would potentially be injected into IPv6 routing in the form of IPv4-mapped addresses.)

In addition, due to the IPv4 address deletion problem, it is desirable that a single IPv4 address needs to be shared via transport port multiplexing technique for different IPv6 nodes when they communicate with other IPv4 hosts.

Furthermore, in the SIIT [RFC2765] implemetation, an IPv6-only node which works through SIIT translators needs some modifications beyond a normal IPv6-only node. These modifications are not strictly implied in this document, since the normal IPv6 addresses can be used in the IPv6 end nodes.

The detailed discussion of the transition scenarios is presented in [FRAMEWORK], the technical specifications of the translation algorithm itself is illustrated in this document.

### 1.1. Translation Model

This document specifies the traslation algorithm that is one of the components descrbed in [FRAMEWORK] needed to make IPv6-only nodes interoperate with IPv4-only nodes as shown in Figure 1.

```
                --------          --------
              //  IPv4  \\      //  IPv6  \\
             /   Domain   \    /   Domain   \
            /             +----+      +--+   \
           |              |XLAT|      |S2|    |  Sn: Servers
           | +--+         +----+      +--+    |  Hn: Clients
           | |S1|         +----+              |
           | +--+         |DNS |      +--+    |  XLAT: V4/V6 Translator
            \     +--+    +----+      |H2|   /   DNS:  DNS Server
             \    |H1|    /    \      +--+  /
              \\  +--+  //      \\        //
                --------          --------
```

Figure 1: Translation Model

The translation model consists of two or more network domains connected by one or more IP/ICMP translators. One of those networks either routes IPv4 but not IPv6, or contains some hosts that only implement IPv4. The other network either routes IPv6 but not IPv4, or contains some hosts that only implement IPv6. Both networks contain clients, servers, and peers.

### 1.2. Applicability and Limitations

The use of this translation algorithm assumes that the IPv6 network is somehow well connected i.e. when an IPv6 node wants to communicate with another IPv6 node there is an IPv6 path between them. Various tunneling schemes exist that can provide such a path, but those mechanisms and their use is outside the scope of this document [RFC2765].

The translation algorithm can be used no only in a subnet or small networks, but can also be used in the autonomous system scope.

The translating function as specified in this document does not translate any IPv4 options and it does not translate IPv6 routing headers, hop-by-hop extension headers, destination options headers or source routing headers [RFC2765].

The issues and algorithms in the translation of datagram containing TCP segments are described in [RFC5382]. The considerations of that document are applicable in this case as well.

Fragmented IPv4 UDP packets that do not contain a UDP checksum (i.e. the UDP checksum field is zero) are not of significant use over wide-areas in the Internet and will not be translated by the IP/ICMP

translator [Miller].

The considerations of The IPSec [RFC4301] [RFC4302] [RFC4303] functionality discussed in [RFC2765] are applicable in this case as well.

IPv4 multicast addresses [RFC3171] can not be mapped to IPv6 multicast addresses [RFC3307] based on the unicast mapping rule. However, special rule of the address translation can be created for the multicast packet translation algorithm and the IP/ICMP header translation aspect of this memo works.

### 1.3. Stateless vs Stateful Mode

The IP/ICMP translator has two possible modes of operation: stateless and stateful. In both cases, we assume that a system 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 in either domain and hence are forced to have their communications translated.

In the stateless mode, one system has an IPv4 address and one has an address of the form specified in [FRAMEWORK], which is explicitly mapped to an IPv4 address. In this mode, there is no need to concern oneself with port translation or translation tables, as the IPv4 and IPv6 counterparts are algorithmically related.

In the stateful mode, the system with the IPv4 address will be represented by that same address type, but the IPv6 system may use any [RFC4291] address except one in that range. In this case, a translation table is required.

### 1.4. IPv4-embedded IPv6 addresses and IPv4-related IPv6 addresses

In SIIT [RFC2765] an IPv6 node should send an IPv6 packet where the destination address is the IPv4-mapped address and the source address is the node's temporarily assigned IPv4-translated address. If the node does not have a temporarily assigned IPv4-translated address it should acquire one. Different from the SIIT model, as described in [FRAMEWORK] the new forms of the IPv6 addresses are introduced.

The IPv4-embedded IPv6 addresses are the IPv6 addresses which have unique relationship to specific IPv4 addresses. This relationship is self described by embedding IPv4 address in the IPv6 address. The IPv4-embedded IPv6 addresses are used for both the statesless and the stateful modes.

The IPv4-related IPv6 addresses are the IPv6 addresses which have

unique relationship to specific IPv4 addresses.  This relationship is maintained as the states (mapping table between IPv4 address/transport_port and IPv6 address/transport_port) in the IP/ICMP translator.  The states are session initiated.  The IPv4-related IPv6 addresses are used fo the stateful mode only.

## 2.  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.  Since the ICMP [RFC0792][RFC4443], TCP [RFC0793] and UDP [RFC0768] headers consist of check sums which include the IP header, the recalculation and updating of the ICMP header and the transport-layer headers MUST be performed.  This is different from [RFC2765], since [RFC2765] uses special prefix (0::ffff:0:a:b:c:d) to avoid the recalculation of the transport-layer header checksum.  The data portion of the packet are left unchanged. The IP/ICMP translator then forwards the packet based on the IPv6 destination address.  The original IPv4 header on the packet is removed and replaced by an IPv6 header.

```
           +-------------+                 +-------------+
           |    IPv4     |                 |    IPv6     |
           |   Header    |                 |   Header    |
           +-------------+                 +-------------+
           |  Transport  |                 |  Fragment   |
           |   Layer     |      ===>       |   Header    |
           |   Header    |                 |(not always) |
           +-------------+                 +-------------+
           |             |                 |  Transport  |
           ~    Data     ~                 |   Layer     |
           |             |                 |   Header    |
           +-------------+                 +-------------+
                                           |             |
                                           ~    Data     ~
                                           |             |
                                           +-------------+
```

Figure 2: IPv4-to-IPv6 Translation

One of the differences between IPv4 and IPv6 is that in IPv6 path MTU discovery is mandatory but it is optional in IPv4.  This implies that IPv6 routers will never fragment a packet - only the sender can do fragmentation.

When the IPv4 node performs path MTU discovery (by setting the DF bit in the header) the path MTU discovery can operate end-to-end i.e. across the translator. In this case either IPv4 or IPv6 routers might send back ICMP "packet too big" messages to the sender. When these ICMP errors are sent by the IPv6 routers they will pass through a translator which will translate the ICMP error to a form that the IPv4 sender can understand. In this case an IPv6 fragment header is only included if the IPv4 packet is already fragmented.

However, when the IPv4 sender does not perform path MTU discovery the translator has to ensure that the packet does not exceed the path MTU on the IPv6 side. This is done by fragmenting the IPv4 packet so that it fits in 1280 byte IPv6 packet since IPv6 guarantees that 1280 byte packets never need to be fragmented. Also, when the IPv4 sender does not perform path MTU discovery the translator MUST always include an IPv6 fragment header to indicate that the sender allows fragmentation. That is needed should the packet pass through an IP/ICMP translator.

The above rules ensure that when packets are fragmented either by the sender or by IPv4 routers that the low-order 16 bits of the fragment identification is carried end-end to ensure that packets are correctly reassembled. In addition, the rules 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 mapping as defined below. Note that ICMP packets require special handling in order to translate the content of ICMP error message and also to add the ICMP pseudo-header checksum.

## 2.1. 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 IPv4 packet MUST be fragmented prior to translating it. Since IPv4 packets with DF not set will always result in a fragment header being added to the packet the IPv4 packets must 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 resulting fragments are then translated independently using the logic described below.

If the DF bit is set and the packet is not a fragment (i.e., the MF flag is not set and the Fragment Offset is zero) then there is no need to add a fragment header to the packet. The IPv6 header fields are set as follows:

Version: 6

Traffic Class: By default, copied from IP Type Of Service and Precedence field (all 8 bits are copied). 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 provide the ability to ignore the IPv4 "TOS" and always set the IPv6 traffic class to zero.

Flow Label: 0 (all zero bits)

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

Next Header: Protocol field copied from IPv4 header

Hop Limit: TTL value copied from 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) needs to check for zero and send the ICMPv4 or ICMPv6 "ttl exceeded" error.

Source Address: The source address is derived from the IPv4 source address to form IPv4-embedded IPv6 address as specified in [FRAMEWORK].

Destination Address: In stateless mode, which is to say that if the IPv4 destination address is within the range of the stateless translation prefix described in Section 1.3, the destination address is derived from the IPv4 destination address to form IPv4-embedded IPv6 address in [FRAMEWORK] [I-D.baker-behave-ivi].

In stateful mode, which is to say that if the IPv4 destination address is not within the range of the stateless translation prefix described in Section 1.3, the IPv6 address (IPv4-related IPv6 address) and transport layer destination port corresponding to the IPv4 destination address and destination port are derived from the database reflecting current session state in the translator [I-D.bagnulo-behave-nat64].

If IPv4 options are present in the IPv4 packet, they are ignored i.e., there is no attempt to translate them. 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 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 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: Protocol field copied from 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.

## 2.2. Translating UDP over IPv4

If a UDP packet has a zero UDP checksum then a valid checksum must be calculated in order to translate the packet. A stateless translator can not do this for fragmented packets but [Miller] indicates that fragmented UDP packets with a zero checksum appear to only be used for malicious purposes. Thus this is not believed to be a noticeable limitation.

When a 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 specifying at least the IP addresses and port numbers in the packet. When it receives fragments other than the first it SHOULD silently drop the packet, since there is no port information to log.

When a translator receives an unfragmented UDP IPv4 packet and the checksum field is zero the translator MUST compute the missing UDP checksum as part of translating the packet. Also, the translator SHOULD maintain a counter of how many UDP checksums are generated in this manner.

## 2.3. Translating ICMPv4 Headers into ICMPv6 Headers

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

In addition all ICMP packets need to have the Type value translated and for ICMP error messages the included IP header also needs translation.

The actions needed to translate various ICMPv4 messages are:

ICMPv4 query messages:

Echo and Echo Reply (Type 8 and Type 0) Adjust the type 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 ICMPv4 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 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) For all that are not explicitly listed below set the Type to 1.

Translate the code field as follows:

Code 0, 1 (net, host unreachable): Set 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 Code to 4 (port unreachable).

Code 4 (fragmentation needed and DF set): Translate to an ICMPv6 Packet Too Big message (Type 2) with code 0. The MTU field needs to be adjusted for the difference between the IPv4 and IPv6 header sizes. Note that if the IPv4 router did not set the MTU field 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.)

Code 5 (source route failed): Set Code to 0 (no route to destination). Note that this error is unlikely since source routes are not translated.

Code 6,7: Set Code to 0 (no route to destination).

Code 8: Set Code to 0 (no route to destination).

Code 9, 10 (communication with destination host administratively prohibited): Set Code to 1 (communication with destination administratively prohibited)

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

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

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

Time Exceeded (Type 11) Set the Type field to 3. The Code field is unchanged.

Parameter Problem (Type 12) Set the Type field to 4. The Pointer needs to be updated to point to the corresponding field in the translated include IP header.

## 2.4. Translating ICMPv4 Error Messages into ICMPv6

There are some differences between the IPv4 and the IPv6 ICMP error message formats as detailed above. In addition, the ICMP error messages contain the IP header for the packet in error which needs to be translated just like a normal IP header. The translation of this "packet in error" is likely to change the length of the datagram thus the Payload Length field in the outer IPv6 header might need to be updated.

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

Figure 3: IPv4-to-IPv6 ICMP Error Translation

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers.

## 2.5. Transport-layer Header Translation

For the IPv6 addresses described in [FRAMEWORK], the recalculation and updating of the transport-layer headers MUST be performed.

## 2.6. Knowing when to Translate

If the IP/ICMP translator is implemented in a router providing both translation and normal forwarding, and the address is reachable by a more specific route without translation, the router should forward it without translating it. Otherwise, when an IP/ICMP translator receives an IPv4 datagram addressed to a destination towards the IPv6 domain, the packet will be translated to IPv6.

## 3. 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 that packet into an IPv4 header. Since the ICMP [RFC0792][RFC4443], TCP [RFC0793] and UDP [RFC0768] headers consist of check sums which include the IP header, the recalculation and updating of the ICMP header and the transport-layer headers MUST be performed. This is different from [RFC2765], since [RFC2765] uses special prefix (0::ffff:0:a:b:c:d) to avoid the recalculation of the transport-layer header checksum. The data portion of the packet are left unchanged. The IP/ICMP translator then forwards the packet based on the IPv4 destination address. The original IPv6 header on the packet is removed and replaced by an IPv4 header.

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

Figure 4: IPv6-to-IPv4 Translation

There are some differences between IPv6 and IPv4 in the area of fragmentation and the minimum link MTU that effect the translation.

An IPv6 link has to have an MTU of 1280 bytes or greater. The corresponding limit for IPv4 is 68 bytes. Thus, unless there were special measures, it would not be possible to do end-to-end path MTU discovery when the path includes an translator since the IPv6 node might receive ICMP "packet too big" messages originated by an IPv4 router that report an MTU less than 1280. However, [RFC2460] requires that IPv6 nodes handle such an ICMP "packet too big" message by reducing the path MTU to 1280 and including an IPv6 fragment header with each packet. This allows end-to-end path MTU discovery across the translator as long as the path MTU is 1280 bytes or greater. When the path MTU drops below the 1280 limit the IPv6 sender will originate 1280 byte packets that will be fragmented by IPv4 routers along the path after being translated to IPv4.

The only drawback with this scheme is that it is not possible to use PMTU to do optimal UDP fragmentation (as opposed to completely avoiding fragmentation) at sender since the presence of an IPv6 Fragment header is interpreted that is it OK to fragment the packet on the IPv4 side. Thus if a UDP application wants to send large packets independent of the PMTU, the sender will only be able to determine the path MTU on the IPv6 side of the translator. If the path MTU on the IPv4 side of the translator is smaller then the IPv6 sender will not receive any ICMP "too big" errors and can not adjust the size fragments it is sending.

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

## 3.1. 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.

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

Identification: All zero.

Flags: The More Fragments flag is set to zero. The Don't Fragments flag is set to one.

Fragment Offset: All zero.

Time to Live: Hop Limit value copied from 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) needs to check for zero and send the ICMPv4 or ICMPv6 "ttl exceeded" error.

Protocol: Next Header field copied from IPv6 header.

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

Source Address: In stateless mode, which is to say that if the IPv6 source address is within the range of the stateless translation prefix described in Section 1.3, the source address is derived from the IPv4-embedded IPv6 address as specified in [FRAMEWORK] [I-D.baker-behave-ivi].

In stateful mode, which is to say that if the IPv6 source address is not within the range of the stateless translation prefix described in Section 1.3, the IPv4 source address and transport layer source port corresponding to the IPv6 source address (IPv4-related IPv6 address) and source port are derived from the database reflecting current session state in the translator as described in [I-D.bagnulo-behave-nat64].

Destination Address: IPv6 packets that are translated have an IPv4-mapped destination address. Thus the address is derived from the IPv6 address as specified in [FRAMEWORK].

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, they are ignored i.e., there is no attempt to translate them. However, the Total Length field and the Protocol field would have to be 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.

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 More Fragments flag is copied from the M flag in the Fragment header. The Don't Fragments flag is set to zero allowing this packet to be fragmented by IPv4 routers.

Fragment Offset: Copied from the Fragment Offset field in the Fragment Header.

Protocol: Next Header value copied from Fragment header.

### 3.2. Translating ICMPv6 Headers into ICMPv4 Headers

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

In addition all ICMP packets need to have the Type value translated and for ICMP error messages the included IP header also needs translation.

The actions needed to translate various ICMPv6 messages are:

ICMPv6 informational messages:

Echo Request and Echo Reply (Type 128 and 129) Adjust the type to 0 and 8, 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 field to 3. Translate the code field as follows:

Code 0 (no route to destination):  Set Code to 1 (host unreachable).

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

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

Code 3 (address unreachable):  Set Code to 1 (host unreachable).

Code 4 (port unreachable):  Set Code to 3 (port unreachable).

Packet Too Big (Type 2)  Translate to an ICMPv4 Destination Unreachable with code 4.  The MTU field needs to 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.

Time Exceeded (Type 3)  Set the Type to 11.  The Code field is unchanged.

Parameter Problem (Type 4)  If the Code is 1 translate this to an ICMPv4 protocol unreachable (Type 3, Code 2).  Otherwise set the Type to 12 and the Code to zero.  The Pointer needs to be updated to point to the corresponding field in the translated include IP header.

Unknown error messages  Silently drop.

### 3.3.  Translating ICMPv6 Error Messages into ICMPv4

There are some differences between the IPv4 and the IPv6 ICMP error message formats as detailed above.  In addition, the ICMP error messages contain the IP header for the packet in error which needs to be translated just like a normal IP header.  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 might need to be updated.

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

Figure 5: IPv6-to-IPv4 ICMP Error Translation

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers.

### 3.4. Transport-layer Header Translation

For the IPv6 addresses described in [FRAMEWORK], the recalculation and updating of the transport-layer headers MUST be performed.

### 3.5. Knowing when to Translate

If the IP/ICMP translator is implemented in a router providing both translation and normal forwarding, and the address is reachable by a more specific route without translation, the router should forward it without translating it. When an IP/ICMP translator receives an IPv6 datagram addressed to a destination towards the IPv4 domain, the packet will be translated to IPv4.

## 4. IANA Considerations

This memo adds no new IANA considerations.

Note to RFC Editor: This section will have served its purpose if it correctly tells IANA that no new assignments or registries are required, or if those assignments or registries are created during

the RFC publication process. From the author's perspective, it may therefore be removed upon publication as an RFC at the RFC Editor's discretion.

## 5. 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 which are used to make the packets reach the translator.

As the Authentication Header [RFC4302] is specified to include the IPv4 Identification field and the translating function not being able to always preserve the Identification field, it is not possible for an IPv6 endpoint to compute AH on received packets that have been translated from IPv4 packets. Thus AH does not work through a translator.

Packets with ESP can be translated since ESP does not depend on header fields prior to the ESP header. Note that ESP transport mode is easier to handle than ESP tunnel mode; in order to use ESP tunnel mode the IPv6 node needs to be able to generate an inner IPv4 header when transmitting packets and remove such an IPv4 header when receiving packets.

## 6. Acknowledgements

This is under development by a large group of people. Those who have posted to the list during the discussion include Andrew Sullivan, Andrew Yourtchenko, Brian Carpenter, Dan Wing, Ed Jankiewicz, Fred Baker, Hiroshi Miyata, Iljitsch van Beijnum, John Schnizlein, Kevin Yin, Magnus Westerlund, Marcelo Bagnulo Braun, Margaret Wasserman, Masahito Endo, Phil Roberts, Philip Matthews, Remi Denis-Courmont, Remi Despres, and Xing Li.

## 7. References

### 7.1. Normative References

[FRAMEWORK]
Baker, F., "Framework for IPv4/IPv6 Translation - baker-behave-v4v6-framework", October 2008.

[I-D.bagnulo-behave-nat64]
Bagnulo, M., Matthews, P., and I. Beijnum, "NAT64: Network

Address and Protocol Translation from IPv6 Clients to IPv4 Servers", draft-bagnulo-behave-nat64-02 (work in progress), November 2008.

[I-D.baker-behave-ivi]
Li, X., Bao, C., Baker, F., and K. Yin, "IVI Update to SIIT and NAT-PT", draft-baker-behave-ivi-01 (work in progress), September 2008.

[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.

[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.

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

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

## 7.2. Informative References

[I-D.petithuguenin-behave-stun-pmtud]
Petit-Huguenin, M., "Path MTU Discovery Using Session Traversal Utilities for NAT (STUN)", draft-petithuguenin-behave-stun-pmtud-02 (work in progress), November 2008.

[Miller] Miller, G., "Email to the ngtrans mailing list", March 1999.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.

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

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[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.

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

[RFC3171] Albanna, Z., Almeroth, K., Meyer, D., and M. Schipper, "IANA Guidelines for IPv4 Multicast Address Assignments", BCP 51, RFC 3171, August 2001.

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

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[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.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

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

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

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

Authors' Addresses

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

Phone: +86 62785983
Email: xing@cernet.edu.cn

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

Phone: +86 62785983
Email: congxiao@cernet.edu.cn

Fred Baker (editor)
Cisco Systems
Santa Barbara, California 93117
USA

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