behave
Internet-Draft
Obsoletes: 2765 (if approved)
Intended status: Standards Track
Expires: August 15, 2010

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

February 11, 2010

IP/ICMP Translation Algorithm
draft-ietf-behave-v6v4-xlate-09

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) for both stateless and stateful modes.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction and Motivation

This document is a product of the 2008-2010 effort to define a replacement for NAT-PT [RFC2766]. 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 transition mechanisms specified in [RFC4213] handle the case of dual IPv4/IPv6 hosts interoperating with both dual IPv4/IPv6 hosts and IPv4-only hosts, which is needed early in the transition to IPv6. The dual IPv4/IPv6 hosts are assigned both one or more IPv4 and one or more IPv6 addresses. The number of available globally unique IPv4 addresses is becoming smaller and smaller as the Internet grows; we expect that there will be a desire to take advantage of the large IPv6 address space and not require that every new Internet node have a permanently and full assigned IPv4 address. Indeed, due to the IPv4 address depletion problem, it is desirable that a single IPv4 address needs to be shared via transport port multiplexing for different IPv6 nodes when they communicate with other IPv4 hosts.

SIIT [RFC2765] was designed for the case of small networks (e.g., a single subnet) and for a site that has IPv6-only hosts in a dual IPv4/IPv6 network. This use case assumes a mechanism for IPv6 nodes to acquire a temporary address from the pool of IPv4 addresses. However, SIIT is not useful in the case when the IPv6 nodes need to acquire temporary IPv4 addresses from a "distant" SIIT box operated by a different administration, or require that the IPv6 Internet contain routes for IPv4-mapped addresses (the latter is considered 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.)

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

A detailed discussion of translation scenarios is presented in [I-D.ietf-behave-v6v4-framework], while the technical specification of the translation algorithm itself is covered in this document.

1.1. IPv4-IPv6 Translation Model

The translation model is consists of two or more network domains connected by one or more IP/ICMP translators (XLATEs) as shown in Figure 1. One of those networks either routes IPv4 but not IPv6, or

contains some hosts that only implement IPv4 or have IPv4-only applications (even if the host and the network support IPv6). The other network either routes IPv6 but not IPv4, or contains some hosts that only implement IPv6 or has IPv6-only applications. Both networks contain clients, servers, and peers. A network domain may also consist of a single host. DNS servers include DNS64 and DNS46, while DNS64 translates A record to AAAA record and DNS46 translates AAAA record to A record.

```
          --------           --------
        //  IPv4  \\       //  IPv6  \\
       /   Domain   \     /   Domain   \
      /             +------+     +--+   \
     |              |XLATE|      |S2|    |  Sn: Servers
     | +--+         +------+     +--+    |  Hn: Clients
     | |S1|         +------+             |
     | +--+         | DNS |      +--+    |  XLATE: IPv4/IPv6 Translator
      \     +--+    +------+     |H2|   /   DNS:   DNS64/DNS46
       \    |H1|    /     \      +--+  /
        \\  +--+  //       \\        //
          --------           --------
```

Figure 1: IPv4-IPv6 Translation Model

The general IPv4/IPv6 translation framework is described in [I-D.ietf-behave-v6v4-framework]. This document specifies the translation algorithms between IPv4 packets and IPv6 packets. The mapping algorithms between IPv4 addresses and IPv6 addresses in the packet headers are specified in [I-D.ietf-behave-address-format].

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. The IPv4 network is also assumed to be well-connected. Various tunneling schemes exist that can provide such paths, but those mechanisms and their use is outside the scope of this document and [RFC2765].

As with [RFC2765], the translating function 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.

The issues and algorithms in the translation of datagrams 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 in the Internet [Miller][Dongjin] and will not be translated by the IP/ICMP translator.

IPv4 multicast addresses [RFC3171] cannot be mapped to IPv6 multicast addresses [RFC3307] based on the unicast mapping rule. However, if multicast address mapping rule is defined, the IP/ICMP header translation aspect of this document works.

1.3. Stateless vs. Stateful Mode

An IP/ICMP translator has two possible modes of operation: stateless and stateful [I-D.ietf-behave-v6v4-framework]. In both cases, we assume that a system (a node or an application) that has an IPv4 address but not an IPv6 address is communicating with a system that has an IPv6 address but no IPv4 address, or that the two systems do not have contiguous routing connectivity and hence are forced to have their communications translated.

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

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

The address translation mechanisms for the stateless and the stateful translations are defined in [I-D.ietf-behave-address-format].

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 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 3 and 4 of this document. The reassembling of fragmented packets in the stateful translator is discussed in [I-D.ietf-behave-v6v4-xlate-stateful], since it requires state maintenance in the translator.

2. 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].

3. 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. Since the ICMPv6 [RFC4443], TCP [RFC0793] and UDP [RFC0768] headers contain checksums that cover IP header information, if the address mapping algorithm is not checksum-neutral, the ICMPv6 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 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

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 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 Don't Fragment (DF) bit, 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 packets, since that is the minimum IPv6 MTU. Also, when the IPv4 sender does not set the DF bit the translator MUST always include an IPv6 fragment header to indicate that the sender allows fragmentation.

The rules in section 3.1 ensure that when packets are fragmented, either by the sender or by IPv4 routers, the low-order 16 bits of the fragment identification are carried end-to-end, ensuring that packets are correctly reassembled. In addition, the rules in section 3.1 use the presence of an IPv6 fragment header to indicate that the sender might not be using path MTU discovery (i.e., the packet should not have the DF flag set should it later be translated back to IPv4).

Other than the special rules for handling fragments and path MTU discovery, the actual translation of the packet header consists of a simple mapping 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.

3.1. Translating IPv4 Headers into IPv6 Headers

If the DF flag is not set and the IPv4 packet will result in an IPv6 packet larger than 1280 bytes, the packet MUST be fragmented so the resulting IPv6 packet (with Fragment header added to each fragment) will be less than or equal to 1280 bytes. For example, if the packet is fragmented prior to the translation, the IPv4 packets 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 MTU of the next-hop interface is less

than the total length value of the IPv4 packet plus 20, the translator MUST send an ICMPv4 "Fragmentation Needed" error message to the IPv4 source address.

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

Version: 6

Traffic Class: By default, copied from 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 provide the ability 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 IPv4 header, minus the size of the IPv4 header and IPv4 options, if present.

Next Header: For ICMPv4 (1) changed to ICMPv6 (58), otherwise protocol field copied from 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) needs to check for zero and send the ICMPv4 "TTL Exceeded" or ICMPv6 "Hop Limit Exceeded" error.

Source Address: The IPv4-converted address derived from the IPv4 source address per [I-D.ietf-behave-address-format] section 2.1.

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

Destination Address: In the stateless mode, which is to say that if the IPv4 destination address is within a range of configured IPv4 stateless translation prefix, the IPv6 destination address is the IPv4-translatable address derived from the IPv4 destination address per [I-D.ietf-behave-address-format] section 2.1. A workflow example of stateless translation is shown in Appendix of this document.

In the stateful mode, which is to say that if the IPv4 destination address is not within the range of any configured IPv4 stateless translation prefix, the IPv6 destination address and corresponding transport-layer destination port are derived from the Binding Information Bases (BIBs) reflecting current session state in the translator as described in [I-D.ietf-behave-v6v4-xlate-stateful].

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

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

IPv6 fields:

Payload Length: Total length value from 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) changed to ICMPv6 (58), otherwise 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.

3.2. 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 need to have the Type value translated and, for ICMPv4 error messages, the included IP header also needs translation.

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 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 codes that are not explicitly listed below, set the Type field to 1, and adjust the ICMP checksum both to take the type change into account and to include the ICMPv6 pseudo-header.

Translate the Code field as follows:

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

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

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

Code 4 (Fragmentation needed and DF set): Translate to an ICMPv6 Packet Too Big message (Type 2) with Code value set to 0. The MTU field needs to be adjusted for the difference between the IPv4 and IPv6 header sizes, i.e. minimum(advertised MTU+20, MTU_of_IPv6_nexthop, (MTU_of_IPv4_nexthop)+20). Note that if the IPv4 router did not set the MTU field (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.)

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

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

Code 8: Set Code value to 0 (No route to destination).

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

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

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

Code 14 (Host Precedence Violation): Silently drop.

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

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 field to 3, and adjust the ICMP checksum both to take the type change into account and to include the ICMPv6 pseudo-header. The Code field is unchanged.

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

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

Other Code values: Silently drop

Unknown ICMPv4 types: Silently drop.

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

Figure 3: Pointer value for translating from IPv4 to IPv6

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

3.3. Translating ICMPv4 Error Messages into ICMPv6

There are some differences between the ICMPv4 and the ICMPv6 error message formats as detailed above. In addition, the ICMP error messages contain for the packet in error, which needs to be translated just like a normal IP packet. The translation of this "packet in error" is likely to change the length of the datagram. Thus the 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 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 SHOULD stop at first embedded header and drop the packet if it contains more.

3.4. Translator Sending ICMPv4 Error Message

If the IPv4 packet is discarded, then the translator SHOULD be able to send back an ICMPv4 error message to the original sender of the packet, unless the discarded packet is itself an ICMPv4 message. The ICMPv4 message, if sent, has a Type value of 3 (Destination Unreachable) and a Code value of 13 (Communication Administratively Prohibited), unless otherwise specified in this document or in [I-D.ietf-behave-v6v4-xlate-stateful]. The translator SHOULD allow an administrator to configure whether the ICMPv4 error messages are sent, rate-limited, or not sent.

3.5. Transport-layer Header Translation

If the address translation algorithm is not checksum neutral, the recalculation and updating of the transport-layer headers MUST be performed.

When a translator receives an unfragmented UDP IPv4 packet and the checksum field is zero, the translator SHOULD 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.

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 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.

For stateful translator, the handling of fragmented UDP IPv4 packets with a zero checksum is discussed in [I-D.ietf-behave-v6v4-xlate-stateful] section 3.1.

3.6. Knowing when to Translate

If the IP/ICMP translator also provides 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.

4. 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], and UDP [RFC0768] headers contain checksums that cover the IP header, if the address mapping algorithm is not checksum-neutral, 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 area 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. Thus, unless there were special measures, it would not be possible to do end-to-end path MTU discovery when the path includes a translator, since the IPv6 node might receive ICMPv6 "Packet Too Big" messages originated by an IPv4 router that report an MTU less than 1280. However, [RFC2460] section 5 requires that IPv6 nodes handle such an ICMPv6 "Packet Too Big" message by reducing the path MTU to 1280 and including an IPv6 fragment header with each packet. In this case, the translator SHOULD set DF to 0 and take the identification value from the IPv6 fragment header when a fragmentation header with (MF=0; Offset=0) is present or set DF to 1 otherwise. 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 drawback with this scheme is that it is not possible to use PMTU discovery to do optimal UDP fragmentation (as opposed to completely avoiding fragmentation) at the sender, since the presence of an IPv6 Fragment header is interpreted that it is okay 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 ICMPv6 "Too Big" errors and cannot adjust the size fragments it is sending.

On the other hand, the recent study indicates that only 43.46% of IPv6-capable web servers include an IPv6 fragmentation header in their respond packets after they were sent an ICMPv6 "Packet Too Big" message specifying an MTU<1280 bytes [Stasiewicz]. A workaround to this problem (ICMPv6 "Packet Too Big" message with MTU<1280) is that if there is no fragmentation header in the IPv6 packet, the translator SHOULD set DF to 0 for the packets equal to or smaller than 1280 bytes and set DF to 1 for packets larger than 1280 bytes. In addition, the translator SHOULD take the identification value from the IPv6 fragmentation header if presents or generate the identification value otherwise. This avoids the introduction of the path MTU discovery black hole. The header translation defined in the next section uses this method.

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 ICMPv6 packets require special handling in order to translate the contents of ICMPv6 error messages and also to remove the ICMPv6 pseudo-header checksum.

4.1. 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 IPv6 header, plus the size of the IPv4 header.

Identification: If the packet size is equal to or smaller than 1280 bytes, generate the identification value. If the packet size is greater than 1280 bytes, set Identification All zeros.

Flags: The More Fragments (MF) flag is set to zero. If the packet size is equal to or smaller than 1280 bytes, the Don't Fragments (DF) flag is set to zero. If the packet size is greater than 1280 bytes, the Don't Fragments (DF) flag 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) needs to check for zero and send the ICMPv4 "TTL Exceeded" or ICMPv6 "Hop Limit Exceeded" error.

Protocol: For ICMPv6 (58) changed to ICMPv4 (1), otherwise Next Header field copied from IPv6 header.

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

Source Address: In the stateless mode, which is to say that if the IPv6 source address is within the range of a configured IPv6 translation prefix, the IPv4 source address is derived from the IPv6 source address per [I-D.ietf-behave-address-format] section 2.1. Note that the original IPv6 source address is an IPv4-translatable address. A workflow example of stateless translation is shown in Appendix of this document. If the translator only supports stateless mode and if the IPv6 source address is not within the range of configured IPv6 prefix(es), the translator SHOULD drop the packet and respond with an ICMPv6 Type=1, Code=5 (Destination Unreachable, Source address failed ingress/egress policy).

In the stateful mode, which is to say that if the IPv6 source address is not within the range of any configured IPv6 stateless translation prefix, the IPv4 source address and transport-layer source port corresponding to the IPv4-related IPv6 source address and source port are derived from the Binding Information Bases (BIBs) as described in [I-D.ietf-behave-v6v4-xlate-stateful].

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

Destination Address: The IPv4 destination address is derived from the IPv6 destination address of the datagram being translated per [I-D.ietf-behave-address-format] section 2.1. Note that the original IPv6 destination address is an IPv4-converted address.

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 is 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 (MF) flag is copied from the M flag in the Fragment header. The Don't Fragments (DF) flag is set to zero allowing this packet to be fragmented if required by IPv4 routers.

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

Protocol: For ICMPv6 (58) changed to ICMPv4 (1), otherwise Next Header field copied from Fragment header.

4.2. Translating ICMPv6 Headers into ICMPv4 Headers

All ICMPv6 messages that are to be translated require that the 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 need to have the Type value translated and, for ICMP error messages, the included IP header also needs translation. Note that the IPv6 addresses in the IPv6 header may not be IPv4-translatable addresses and there will be no corresponding IPv4 addresses represented of this IPv6 address. In this case, the translator can ether do stateful translation or map them to an IPv4 address block as a holder for all non IPv4-translatable IPv6 addresses in a stateless manner.

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 field to 3, and adjust the ICMP checksum both to take the type change into account and to exclude the ICMPv6 pseudo-header. Translate the Code field as follows:

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

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

Code 2 (Beyond scope of source address): Set Code value 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 Code value to 1 (Host unreachable).

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

Other Code values: Silently drop.

Packet Too Big (Type 2): Translate to an ICMPv4 Destination Unreachable (Type 3) with Code value equal to 4, and adjust the ICMPv4 checksum both to take the type change into account and to exclude the ICMPv6 pseudo-header. 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, i.e. minimum(advertised MTU-20, MTU_of_IPv4_nexthop, (MTU_of_IPv6_nexthop)-20)

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

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

Code 0 (Erroneous header field encountered): Set 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 those containing IPv6 address literals will not have those addresses translated to IPv4 address literals; this may cause problems with processing of those ICMP extensions.

4.3. Translating ICMPv6 Error Messages into ICMPv4

There are some differences between the ICMPv4 and the ICMPv6 error message formats as detailed above. In addition, the ICMP error messages contain for the packet in error, which needs to be translated just like a normal IP packet. The translation of this "packet in error" is likely to change the length of the datagram thus the Total Length field in the outer IPv4 header 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 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 SHOULD stop at first embedded header and drop the packet if it contains more. Note that the IPv6 addresses in the IPv6 header may not be IPv4-translatable addresses and there will be no corresponding IPv4 addresses. In this case, the translator can ether do stateful translation or map them to an IPv4 address block as a holder for all non IPv4-translatable IPv6 addresses.

4.4. Translator Sending ICMPv6 Error Message

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

If the reason of sending ICMPv6 is due to that the IPv6 source address is not an IPv4-translatable address and the translator is stateless, the ICMPv6 message, if sent, has a Type value 1 and Code value 5 (Source address failed ingress/egress policy). In other cases, the ICMPv6 message has a Type value of 1 (Destination Unreachable) and a Code value of 1 (Communication with destination administratively prohibited), unless otherwise specified in this document or [I-D.ietf-behave-v6v4-xlate-stateful]. The translator SHOULD allow an administrator to configure whether the ICMPv6 error messages are sent, rate-limited, or not sent.

4.5. Transport-layer Header Translation

If the address translation algorithm is not checksum neutral, the recalculation and updating of the transport-layer headers MUST be performed.

4.6. Knowing when to Translate

If the IP/ICMP translator also provides 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 an host in IPv4 domain, the IPv6 packet MUST be translated to IPv4.

5. 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.

6. 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. The [I-D.ietf-behave-address-format] addresses these issues.

As the Authentication Header [RFC4302] is specified to include the IPv4 Identification field and the translating function is not able to always preserve the Identification field, it is not possible for an IPv6 endpoint to verify the 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.

7. 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, Dave Thaler, Ed Jankiewicz, Hiroshi Miyata, Iljitsch van Beijnum, Jari Arkko, Jerry Huang, John Schnizlein, Jouni Korhonen, Kentaro Ebisawa, Kevin Yin, Magnus Westerlund, Marcelo Bagnulo Braun, Margaret Wasserman, Masahito Endo, Phil Roberts, Philip Matthews, Remi Denis-Courmont, Remi Despres, Senthil Sivakumar, Simon Perreault and Zen Cao.

8. Appendix: Stateless translation workflow example

A stateless translation workflow example is depicted in the following figure:

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

Figure 8

A translator (XLATE) connects the IPv6 network to the IPv4 network. This XLATE uses the Network Specific Prefix (NSP) 2001:DB8:100::/40 defined in [I-D.ietf-behave-address-format] 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 166.111.1.2.

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 XLATE.

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 XLATE.

8.1. 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:DB8:1a6:6f01:200:: is formed from 166.111.1.2 based on the address mapping algorithm [I-D.ietf-behave-address-format].

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:DB8:1a6:6f01:200::.

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

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

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

   * The XLATE includes 192.0.2.33 as source address in the packet and 166.111.1.2 as destination address in the packet. Note that 192.0.2.33 and 166.111.1.2 are extracted directly from the source IPv6 address 2001:DB8:1C0:2:21:: (IPv4-translatable address) and destination IPv6 address 2001:DB8:1a6:6f01:200:: (IPv4-converted address) of the received IPv6 packet that is being translated.

5. The XLATE sends the translated packet out 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 166.111.1.2.

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

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

   * The XLATE includes 2001:DB8:1C0:2:21:: as destination address in the packet and 2001:DB8:1a6:6f01:200:: as source address in the packet. Note that 2001:DB8:1C0:2:21:: and 2001:DB8:1a6:6f01:200:: are formed directly from the destination IPv4 address 192.0.2.33 and source IPv4 address 166.111.1.2 of the received IPv4 packet that is being translated.

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

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

8.2. H4 establishes communication with H6

The steps by which H4 establishes communication with H6 are:

1. H4 performs the destination address mapping, so 192.0.2.33 is formed from IPv4-translatable address 2001:DB8:1C0:2:21:: based on the address mapping algorithm [I-D.ietf-behave-address-format].

2. H4 sends a packet to H6. The packet is sent from a source address 166.111.1.2 to a destination address 192.0.2.33.

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

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

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

   * The XLATE includes 2001:DB8:1a6:6f01:200:: as source address in the packet and 2001:DB8:1C0:2:21:: as destination address in the packet. Note that 2001:DB8:1a6:6f01:200:: (IPv4-converted address) and 2001:DB8:1C0:2:21:: (IPv4-translatable address) are obtained directly from the source IPv4 address 166.111.1.2 and destination IPv4 address 192.0.2.33 of the received IPv4 packet that is being translated.

5. The XLATE 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:DB8:1a6:6f01:200:: and source address 2001:DB8:1C0:2:21::.

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

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

   * The XLATE includes 166.111.1.2 as destination address in the packet and 192.0.2.33 as source address in the packet. Note that 166.111.1.2 and 192.0.2.33 are formed directly from the destination IPv6 address 2001:DB8:1a6:6f01:200:: 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 finished.

9. References

9.1. Normative References

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

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

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

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

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

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

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

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

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

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

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

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

9.2. Informative References

[Dongjin] Lee, D., "Email to the behave mailing list (http://www.ietf.org/mail-archive/web/behave/current/msg06856.html)", Sept 2009.

[I-D.ietf-behave-address-format]
Huitema, C., Bao, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", draft-ietf-behave-address-format-04 (work in progress), January 2010.

[I-D.ietf-behave-v6v4-framework]
Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", draft-ietf-behave-v6v4-framework-06 (work in progress), February 2010.

[I-D.ietf-behave-v6v4-xlate-stateful]
Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-v6v4-xlate-stateful-08 (work in progress), January 2010.

[Miller] Miller, G., "Email to the ngtrans mailing list (http://www.mail-archive.com/ipv6@ietf.org/msg10159.html)", March 1999.

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

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

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

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

[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.

[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.

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

[Stasiewicz]
Stasiewicz, B., "Email to the behave mailing list (http://www.ietf.org/mail-archive/web/behave/current/msg08093.html)", Feb 2010.

Authors' Addresses

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

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

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

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

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA

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