v6ops
Internet-Draft
Obsoletes: 6145 (if approved)
Intended status: Standards Track
Expires: October 28, 2016

C. Bao
X. Li
CERNET Center/Tsinghua University
F. Baker
Cisco Systems
T. Anderson
Redpill Linpro
F. Gont
Huawei Technologies

April 26, 2016

IP/ICMP Translation Algorithm (rfc6145bis)
draft-bao-v6ops-rfc6145bis-07

Abstract

This document describes the Stateless IP/ICMP Translation Algorithm (SIIT), which translates between IPv4 and IPv6 packet headers (including ICMP headers). This document obsoletes RFC 6145.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 28, 2016.

Table of Contents

1. Introduction and Motivation 3
1.1. IPv4-IPv6 Translation Model 3
1.2. Applicability and Limitations 3
1.3. Stateless vs. Stateful Mode 4
1.4. Path MTU Discovery and Fragmentation 5
2. Changes from RFC 6145 5
3. Conventions 5
4. Translating from IPv4 to IPv6 6
4.1. Translating IPv4 Headers into IPv6 Headers 7
4.2. Translating ICMPv4 Headers into ICMPv6 Headers 9
4.3. Translating ICMPv4 Error Messages into ICMPv6 13
4.4. Generation of ICMPv4 Error Message 14
4.5. Transport-Layer Header Translation 14
4.6. Knowing When to Translate 15
5. Translating from IPv6 to IPv4 15
5.1. Translating IPv6 Headers into IPv4 Headers 17
5.1.1. IPv6 Fragment Processing 19
5.2. Translating ICMPv6 Headers into ICMPv4 Headers 19
5.3. Translating ICMPv6 Error Messages into ICMPv4 22
5.4. Generation of ICMPv6 Error Messages 22
5.5. Transport-Layer Header Translation 23
5.6. Knowing When to Translate 23
6. Mapping of IP Addresses 23
7. Special Considerations for ICMPv6 Packet Too Big 24
8. IANA Considerations 24
9. Security Considerations 24
10. Acknowledgements 24
11. References 25
11.1. Normative References 25
11.2. Informative References 28
Appendix A. Stateless Translation Workflow Example 33
A.1. H6 Establishes Communication with H4 34
A.2. H4 Establishes Communication with H6 35

1. Introduction and Motivation

This document obsoletes [RFC6145].

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

1.1. IPv4-IPv6 Translation Model

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

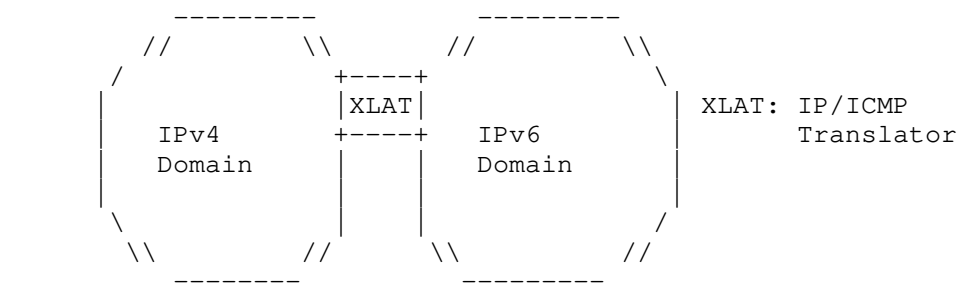


Figure 1: IPv4-IPv6 Translation Model

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

1.2. Applicability and Limitations

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

As with [RFC6145], the translating function specified in this document does not translate any IPv4 options, and it does not translate IPv6 extension headers except the Fragment Header.

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

Fragmented IPv4 UDP packets that do not contain a UDP checksum (i.e., the UDP checksum field is zero) are not of significant use in the Internet, and in general will not be translated by the IP/ICMP translator [Section 4.5]. However, when the translator is configured to forward the packet without a UDP checksum, the fragmented IPv4 UDP packets will be translated.

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

The IP/ICMP header translation specified in this document is consistent with requirements of multicast IP/ICMP headers. However, IPv4 multicast addresses [RFC5771] cannot be mapped to IPv6 multicast addresses [RFC3307] based on the unicast mapping rule [RFC6052]. An example of experiments of the multicast address mapping can be found in [RFC6219].

1.3. Stateless vs. Stateful Mode

An IP/ICMP translator has two possible modes of operation: stateless and stateful [RFC6144]. In both cases, we assume that a system (a node or an application) that has an IPv4 address but not an IPv6 address is communicating with a system that has an IPv6 address but no IPv4 address, or that the two systems do not have contiguous routing connectivity, or they might have contiguous routing connectivity but are interacting via masking addresses (i.e. hairpinning) [RFC4787] and hence are forced to have their communications translated.

In the stateless mode, an IP/ICMP translator will convert IPv4 addresses to IPv6 and vice versa solely based on the configuration of the stateless IP/ICMP translator and information contained within the packet being translated. For example, for the default behavior defined in [RFC6052], a specific IPv6 address range will represent IPv4 systems (IPv4-converted addresses), and the IPv6 systems have addresses (IPv4-translatable addresses) that can be algorithmically mapped to a subset of the service provider's IPv4 addresses. Other stateless translation algorithms are defined in Section 6. The stateless translator does not keep any dynamic session or binding state, thus there is no requirement that the packets in a single session or flow traverses a single translator.

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

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

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

1.4. Path MTU Discovery and Fragmentation

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

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

2. Changes from RFC 6145

The changes from RFC 6145 are the following:

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

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

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

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

3. Conventions

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

4. Translating from IPv4 to IPv6

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

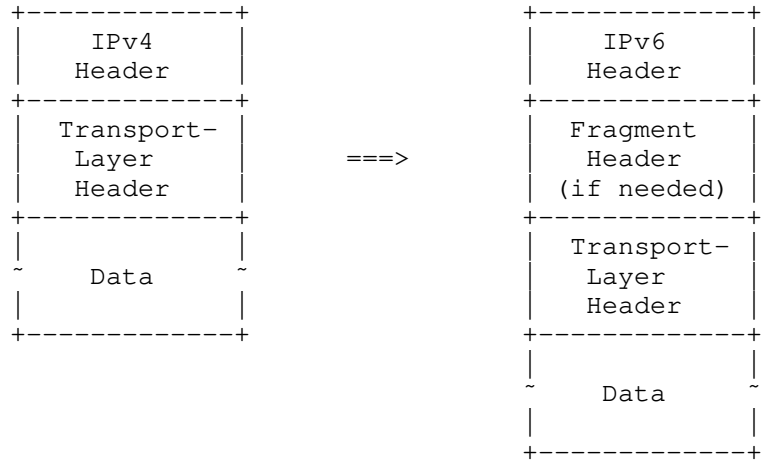


Figure 2: IPv4-to-IPv6 Translation

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

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

However, when the IPv4 sender does not set the DF bit, the translator MUST ensure that the packet does not exceed the path MTU on the IPv6 side. This is done by fragmenting the IPv4 packet (with Fragment Headers) so that it fits in 1280-byte IPv6 packets, since that is the

minimum IPv6 MTU. The IPv6 Fragment Header has been shown to cause operational difficulties in practice due to limited firewall fragmentation support, etc. In an environment where the network owned/operated by the same entity that owns/operates the translator, the translator MUST provide a configuration function for the network administrator to adjust the threshold of the minimum IPv6 MTU to a value that reflects the real value of the minimum IPv6 MTU in the network (greater than 1280 bytes). This will help reduce the chance of including the Fragment Header in the packets.

When the IPv4 sender does not set the DF bit, the translator MUST NOT include the Fragment Header for the non-fragmented IPv6 packets.

The rules in Section 4.1 ensure that when packets are fragmented, either by the sender or by IPv4 routers, the low-order 16 bits of the fragment identification are carried end-to-end, ensuring that packets are correctly reassembled.

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

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

4.1. Translating IPv4 Headers into IPv6 Headers

If the DF flag is not set and the IPv4 packet will result in an IPv6 packet larger than a user-defined length (hereinafter referred to as "lowest-ipv6-mtu", and which defaults to 1280 bytes), the packet SHOULD be fragmented so the resulting IPv6 packet (with Fragment Header added to each fragment) will be less than or equal to lowest-ipv6-mtu, For example, if the packet is fragmented prior to the translation, the IPv4 packets should be fragmented so that their length, excluding the IPv4 header, is at most 1232 bytes (1280 minus 40 for the IPv6 header and 8 for the Fragment Header). The translator MUST provide a configuration function for the network administrator to adjust the threshold of the minimum IPv6 MTU to a value greater than 1280-byte if the real value of the minimum IPv6 MTU in the network is known to the administrator. The resulting fragments are then translated independently using the logic described below.

If the DF bit is set and the MTU of the next-hop interface is less than the total length value of the IPv4 packet plus 20, the

translator MUST send an ICMPv4 "Fragmentation Needed" error message to the IPv4 source address.

The IPv6 header fields are set as follows:

Version: 6

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

Flow Label: 0 (all zero bits)

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

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

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

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

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

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

If any IPv4 options are present in the IPv4 packet, they MUST be

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

If there is a need to add a Fragment Header (the packet is a fragment or the DF bit is not set and the packet size is greater than the minimum IPv6 MTU in the network set by the translator configuration function), the header fields are set as above with the following exceptions:

IPv6 fields:

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

Next Header: Fragment Header (44).

Fragment Header fields:

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

Fragment Offset: Fragment Offset copied from the IPv4 header.

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

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

4.2. Translating ICMPv4 Headers into ICMPv6 Headers

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

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

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

ICMPv4 query messages:

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

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

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

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

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

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

Unknown ICMPv4 types: Silently drop.

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

ICMPv4 error messages:

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

Translate the Code as follows:

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

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

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

Code 4 (Fragmentation Needed and DF was Set): Translate to an ICMPv6 Packet Too Big message (Type 2) with Code set to 0. The MTU field MUST be adjusted for the difference between the IPv4 and IPv6 header sizes, but MUST NOT be set to a value smaller than the minimum IPv6 MTU (1280 bytes). That is, it should be set to maximum(1280, minimum((MTU value in the Packet Too Big Message)+20, MTU_of_IPv6_nexthop, (MTU_of_IPv4_nexthop)+20)). Note that if the IPv4 router set the MTU field to zero, i.e., the router does not implement [RFC1191], then the translator MUST use the plateau values specified in [RFC1191] to determine a likely path MTU and include that path MTU in the ICMPv6 packet. (Use the greatest plateau value that is less than the returned Total Length field, but that is larger than or equal to 1280.)

See also the requirements in Section 7.

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

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

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

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

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

Code 14 (Host Precedence Violation): Silently drop.

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

Other Code values: Silently drop.

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

Alternative Host Address (Type 6): Silently drop.

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

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

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

Translate the Code as follows:

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

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

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

Other Code values: Silently drop.

Unknown ICMPv4 types: Silently drop.

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

Figure 3: Pointer Value for Translating from IPv4 to IPv6

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

4.3. Translating ICMPv4 Error Messages into ICMPv6

There are some differences between the ICMPv4 and the ICMPv6 error message formats as detailed above. The ICMP error messages containing the packet in error MUST be translated just like a normal IP packet (except the TTL value of the inner IPv4/IPv6 packet). If the translation of this "packet in error" changes the length of the datagram, the Total Length field in the outer IPv6 header MUST be updated.

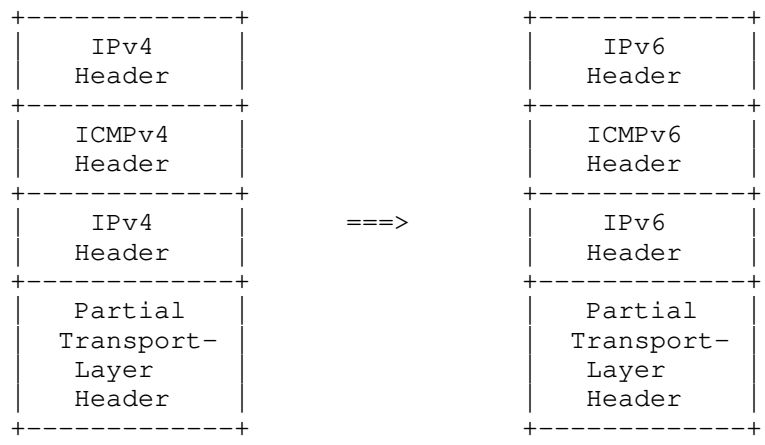


Figure 4: IPv4-to-IPv6 ICMP Error Translation

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

4.4. Generation of ICMPv4 Error Message

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

4.5. Transport-Layer Header Translation

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

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

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

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

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

   For a stateful translator, the handling of fragmented UDP IPv4 packets with a zero checksum is discussed in [RFC6146], Section 3.4.

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

4.6. Knowing When to Translate

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

5. Translating from IPv6 to IPv4

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

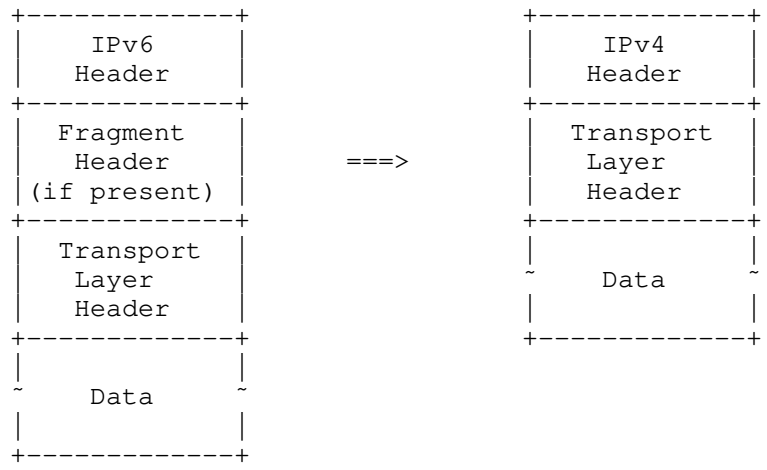


Figure 5: IPv6-to-IPv4 Translation

There are some differences between IPv6 and IPv4 (in the areas of fragmentation and the minimum link MTU) that affect the translation. An IPv6 link has to have an MTU of 1280 bytes or greater. The corresponding limit for IPv4 is 68 bytes. Path MTU discovery across a translator relies on ICMP Packet Too Big messages being received and processed by IPv6 hosts.

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

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

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

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

Other than the special rules for handling fragments and path MTU

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

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

5.1. Translating IPv6 Headers into IPv4 Headers

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

Version: 4

Internet Header Length: 5 (no IPv4 options)

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

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

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

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

Fragment Offset: All zeros.

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

ICMPv6 "Hop Limit Exceeded" error.

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

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

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

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

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

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

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

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

5.1.1. IPv6 Fragment Processing

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

Total Length: If the Next Header field of the Fragment Header is an extension header (except ESP, but including AH) then the packet SHOULD be dropped and logged. For other cases, the Total Length MUST be set to Payload Length value from IPv6 header, minus length of extension headers up to Fragmentation Header, minus 8 for the Fragment Header, plus the size of the IPv4 header.

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

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

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

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

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

5.2. Translating ICMPv6 Headers into ICMPv4 Headers

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

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

The actions needed to translate various ICMPv6 messages are:

ICMPv6 informational messages:

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

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

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

Unknown informational messages: Silently drop.

ICMPv6 error messages:

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

Translate the Code as follows:

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

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

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

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

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

Other Code values: Silently drop.

Packet Too Big (Type 2): Translate to an ICMPv4 Destination Unreachable (Type 3) with Code 4, and adjust the ICMPv4 checksum both to take the type change into account and to exclude the ICMPv6 pseudo-header. The MTU field MUST be adjusted for the difference between the IPv4 and IPv6 header

sizes, taking into account whether or not the packet in error includes a Fragment Header, i.e., minimum((MTU value in the Packet Too Big Message)-20, MTU_of_IPv4_nexthop, (MTU_of_IPv6_nexthop)-20).

See also the requirements in Section 7.

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

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

Translate the Code as follows:

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

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

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

Unknown error messages: Silently drop.

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

Figure 6: Pointer Value for Translating from IPv6 to IPv4

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

5.3. Translating ICMPv6 Error Messages into ICMPv4

There are some differences between the ICMPv4 and the ICMPv6 error message formats as detailed above. The ICMP error messages containing the packet in error MUST be translated just like a normal IP packet (except that the TTL/Hop Limit value of the inner IPv4/IPv6 packet are not decremented). The translation of this "packet in error" is likely to change the length of the datagram; thus, the Total Length field in the outer IPv4 header MUST be updated.

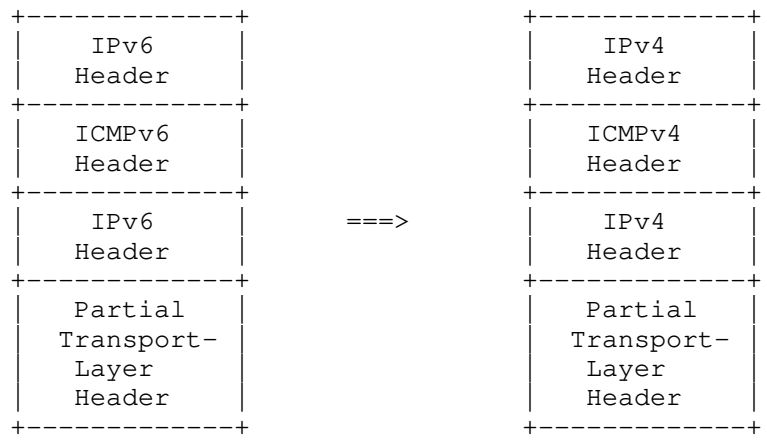


Figure 7: IPv6-to-IPv4 ICMP Error Translation

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

5.4. Generation of ICMPv6 Error Messages

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

the discarded packet is itself an ICMPv6 message.

The ICMPv6 message MUST have Type 1 (Destination Unreachable) and Code 1 (Communication with destination administratively prohibited), unless otherwise specified in this document or [RFC6146]. The translator SHOULD allow an administrator to configure whether the ICMPv6 error messages are sent, rate-limited, or not sent.

5.5. Transport-Layer Header Translation

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

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

5.6. Knowing When to Translate

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

6. Mapping of IP Addresses

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

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

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

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

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

7. Special Considerations for ICMPv6 Packet Too Big

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

8. IANA Considerations

This document makes no requests of the IANA.

9. Security Considerations

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

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

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

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

10. Acknowledgements

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

11. References

11.1. Normative References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

11.2. Informative References

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

[I-D.ietf-6man-deprecate-atomfrag-generation] Gont, F., LIU, S., and T. Anderson, "Generation of IPv6 Atomic Fragments Considered Harmful", draft-ietf-6man-deprecate-atomfrag-generation-06 (work in progress), April 2016.

[I-D.ietf-6man-rfc2460bis] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", draft-ietf-6man-rfc2460bis-04 (work in progress), March 2016.

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

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

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <http://www.rfc-editor.org/info/rfc2474>.

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

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

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

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

[RFC3590] Haberman, B., "Source Address Selection for the Multicast Listener Discovery (MLD) Protocol", RFC 3590, DOI 10.17487/RFC3590, September 2003, <http://www.rfc-editor.org/info/rfc3590>.

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

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

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

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <http://www.rfc-editor.org/info/rfc4787>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <http://

www.rfc-editor.org/info/rfc4821>.

[RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", RFC 5737, DOI 10.17487/RFC5737, January 2010, <http://www.rfc-editor.org/info/rfc5737>.

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

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

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

[RFC7599] Li, X., Bao, C., Dec, W., Ed., Troan, O., Matsushima, S., and T. Murakami, "Mapping of Address and Port using Translation (MAP-T)", RFC 7599, DOI 10.17487/RFC7599, July 2015, <http://www.rfc-editor.org/info/rfc7599>.

Appendix A. Stateless Translation Workflow Example

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

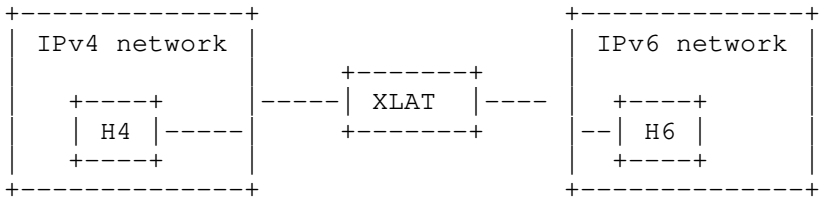


Figure 8

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

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

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

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

IPv4 interface of the XLAT.

A.1. H6 Establishes Communication with H4

The steps by which H6 establishes communication with H4 are:

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

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

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

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

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

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

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

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

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

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

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

198.51.100.2 of the received IPv4 packet that is being translated.

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

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

A.2. H4 Establishes Communication with H6

The steps by which H4 establishes communication with H6 are:

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

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

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

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

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

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

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

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

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA

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

Tore Anderson
Redpill Linpro
Vitaminveien 1A
0485 Oslo
Norway

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

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

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

IPv6 Operations Working Group (v6ops)
Internet-Draft
Intended status: Informational
Expires: September 12, 2016

F. Gont
SI6 Networks / UTN-FRH
N. Hilliard
INEX
G. Doering
SpaceNet AG
W. Liu
Huawei Technologies
W. Kumari
Google

March 11, 2016

Operational Implications of IPv6 Packets with Extension Headers
draft-gont-v6ops-ipv6-ehs-packet-drops-03

Abstract

This document summarizes the security and operational implications of IPv6 extension headers, and attempts to analyze reasons why packets with IPv6 extension headers may be dropped in the public Internet.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2016.

Copyright Notice

Table of Contents

1. Introduction 2
2. Previous Work on IPv6 Extension Headers 3
3. Security Implications 4
4. Operational Implications 5
4.1. Requirement to process required layer-3/layer-4 information 5
4.2. Route-Processor Protection 7
4.3. Inability to Perform Fine-grained Filtering 8
5. A Possible Attack Vector 8
6. Future Work 10
7. IANA Considerations 10
8. Security Considerations 10
9. Acknowledgements 11
10. References 11
10.1. Normative References 11
10.2. Informative References 12
Authors' Addresses 15

1. Introduction

IPv6 Extension Headers (EHs) allow for the extension of the IPv6 protocol, and provide support for core functionality such as IPv6 fragmentation. However, common implementation limitations suggest that EHs present a challenge for IPv6 packet routing equipment, and evidence exists to suggest that IPv6 packets with EHs may be intentionally dropped on the public Internet in some network deployments.

The authors of this document have been involved in numerous discussions about IPv6 extension headers (both within the IETF and outside of it), and have noticed that a number of security and operational issues were unknown to the larger audience participating in these discussions.

This document has the following goals:

- Raise awareness about the security and operational implications of IPv6 Extension Headers, and presents reasons why some networks intentionally drop packets containing IPv6 Extension Headers.

- Highlight areas where current IPv6 support by networking devices maybe sub-optimal, such that the aforementioned support is improved.

- Highlight operational issues associated with IPv6 extension headers, such that those issues are considered in IETF standardization efforts.

Section 2 of this document summarizes the previous work that has been done in the area of IPv6 extension headers. Section 3 briefly discusses the security implications of IPv6 Extension Headers, while Section 4 discusses their operational implications. Finally, Section 6 proposes an action plan for improving the state of affairs of IPv6 extension headers.

2. Previous Work on IPv6 Extension Headers

Some of the implications of IPv6 Extension Headers have been discussed in IETF circles. For example, [I-D.taylor-v6ops-fragdrop] discusses a rationale for which operators drop IPv6 fragments. [I-D.wkumari-long-headers] discusses possible issues arising from "long" IPv6 header chains. [RFC7045] clarifies how intermediate nodes should deal with IPv6 extension headers. [RFC7112] discusses the issues arising in a specific fragmentation case where the IPv6 header chain is fragmented into two or more fragments (and formally forbids such fragmentation case). [I-D.kampanakis-6man-ipv6-eh-parsing] describes how inconsistencies in the way IPv6 packets with extension headers are parsed by different implementations may result in evasion of security controls, and presents guidelines for parsing IPv6 extension headers with the goal of providing a common and consistent parsing methodology for IPv6 implementations. [RFC6980] analyzes the security implications of employing IPv6 fragmentation with Neighbor Discovery for IPv6, and formally recommends against such usage. Finally, [RFC7123] discusses how some popular RA-Guard implementations are subject to evasion by means of IPv6 extension headers.

Some preliminary measurements regarding the extent to which packet containing IPv6 EHs are dropped in the public Internet have been presented in [PMTUD-Blackholes], [Gont-IEPG88], [Gont-Chown-IEPG89], and [Linkova-Gont-IEPG90]. [I-D.ietf-v6ops-ipv6-ehs-in-real-world] presents more comprehensive results and documents the methodology for obtaining the presented results.

3. Security Implications

The security implications of IPv6 Extension Headers generally fall into one or more of these categories:

- Evasion of security controls
- DoS due to processing requirements
- DoS due to implementation errors
- Extension Header-specific issues

Unlike IPv4 packets where the upper-layer protocols can be trivially found by means of the "IHL" ("Internet Header Length") IPv4 header field, the structure of IPv6 packets is more flexible and complex. Locating upper-layer protocol information requires that all IPv6 extension headers be examined. This has presented implementation difficulties, and packet filtering mechanisms that require upper-layer information (even if just the upper layer protocol type) on several security devices can be trivially evaded by inserting IPv6 Extension Headers between the main IPv6 header and the upper layer protocol. [RFC7113] describes this issue for the RA-Guard case, but the same techniques can be employed to circumvent other IPv6 firewall and packet filtering mechanisms. Additionally, implementation inconsistencies in packet forwarding engines may result in evasion of security controls [I-D.kampanakis-6man-ipv6-eh-parsing] [Atlasis2014] [BH-EU-2014].

Packets that use IPv6 Extension Headers may have a negative performance impact on the handling devices. Unless appropriate mitigations are put in place (e.g., packet dropping and/or rate-limiting), an attacker could simply send a large amount of IPv6 traffic employing IPv6 Extension Headers with the purpose of performing a Denial of Service (DoS) attack (see Section 4 for further details).

NOTE:
In the most trivial case, a packet that includes a Hop-by-Hop Options header will typically go through the slow forwarding path, and be processed by the router's CPU. Another possible case might be that in which a router that has been configured to enforce an ACL based on upper-layer information (e.g., upper layer protocol or TCP Destination Port), needs to process the entire IPv6 header chain (in order to find the required information) and this causes the packet to be processed in the slow path [Cisco-EH-Cons]. We note that, for obvious reasons, the aforementioned performance issues may also affect other devices such as firewalls, Network

Intrusion Detection Systems (NIDS), etc. [Zack-FW-Benchmark]. The extent to which these devices are affected will typically be implementation-dependent.

IPv6 implementations, like all other software, tend to mature with time and wide-scale deployment. While the IPv6 protocol itself has existed for almost 20 years, serious bugs related to IPv6 Extension Header processing continue to be discovered. Because there is currently little operational reliance on IPv6 Extension headers, the corresponding code paths are rarely exercised, and there is the potential that bugs still remain to be discovered in some implementations.

IPv6 Fragment Headers are employed to allow fragmentation of IPv6 packets. While many of the security implications of the fragmentation / reassembly mechanism are known from the IPv4 world, several related issues have crept into IPv6 implementations. These range from denial of service attacks to information leakage, for example [I-D.ietf-6man-predictable-fragment-id], [Bonica-NANOG58] and [Atlasis2012]).

4. Operational Implications

4.1. Requirement to process required layer-3/layer-4 information

Intermediate systems and middleboxes that need to find the layer-4 header must process the entire IPv6 extension header chain. When such devices are unable to obtain the required information, they may simply drop the corresponding packets. The following subsections discuss some of reasons for which such layer-4 information may be needed by an intermediate systems or middlebox, and why packets containing IPv6 extension headers may represent a challenge in such scenarios.

4.1.1. Packet Forwarding Engine Constraints

Most modern routers use dedicated hardware (e.g. ASICs or NPUs) to determine how to forward packets across their internal fabrics (see [IEPG94-Scudder] for details). One of the common methods of handling next-hop lookup is to send a small portion of the ingress packet to a lookup engine with specialised hardware (e.g. ternary CAM or RLDRAM) to determine the packet's next-hop. Technical constraints mean that there is a trade-off between the amount of data sent to the lookup engine and the overall performance of the lookup engine. If more data is sent, the lookup engine can inspect further into the packet, but the overall performance of the system will be reduced. If less data is sent, the overall performance of the router will be increased

but the packet lookup engine may not be able to inspect far enough into a packet to determine how it should be handled.

NOTE:
For example, current high-end routers at the time of authorship of this document can use up to 192 bytes of header (Cisco ASR9000 Typhoon) or 384 bytes of header (Juniper MX Trio)

If a hardware forwarding engine on a modern router cannot make a forwarding decision about a packet because critical information is not sent to the look-up engine, then the router will normally drop the packet. Historically, some packet forwarding engines punted packets of this form to the control plane for more in-depth analysis, but this is unfeasible on most current router architectures as a result of the vast difference between the hardware forwarding capacity of the router and processing capacity of the control plane and the size of the management link which connects the control plane to the forwarding plane.

If an IPv6 header chain is sufficiently long that its header exceeds the packet look-up capacity of the router, then it may be dropped due to hardware inability to determine how it should be handled.

4.1.2. ECMP and Hash-based Load-Sharing

In the case of ECMP (equal cost multi path) load sharing, the router on the sending side of the link needs to make a decision regarding which of the links to use for a given packet. Since round-robin usage of the links is usually avoided in order to prevent packet reordering, forwarding engines need to use a mechanism which will consistently forward the same data streams down the same forwarding paths. Most forwarding engines achieve this by calculating a simple hash using an n-tuple gleaned from a combination of layer-2 through to layer-4 packet header information. This n-tuple will typically use the src/dst MAC address, src/dst IP address, and if possible further layer-4 src/dst port information. As layer-4 port information increases the entropy of the hash, it is highly desirable to use it where possible.

We note that in the IPv6 world, flows are expected to be identified by means of the IPv6 Flow Label [RFC6437]. Thus, ECMP and Hash-based Load-Sharing would be possible without the need to process the entire IPv6 header chain to obtain upper-layer information to identify flows. However, we note that for a long time many IPv6 implementations failed to set the Flow Label, and ECMP and Hash-based Load-Sharing devices also did not employ the Flow Label for performing their task.

Clearly, widespread support of [RFC6437] would relieve middle-boxes from having to process the entire IPv6 header chain, making Flow Label-based ECMP and Hash-based Load-Sharing [RFC6438] feasible.

4.1.3. Enforcing infrastructure ACLs

Generally speaking, infrastructure ACLs (iACLs) drop unwanted packets destined to parts of a provider's infrastructure, because they are not operationally needed and can be used for attacks of different sorts against the router's control plane. Some traffic needs to be differentiated depending on layer-3 or layer-4 criteria to achieve a useful balance of protection and functionality, for example:

- Permit some amount of ICMP echo (ping) traffic towards the router's addresses for troubleshooting.

- Permit BGP sessions on the shared network of an exchange point (potentially differentiating between the amount of packets/seconds permitted for established sessions and connection establishment), but do not permit other traffic from the same peer IP addresses.

4.1.4. DDoS Management and Customer Requests for Filtering

The case of customer DDoS protection and edge-to-core customer protection filters is similar in nature to the infrastructure ACL protection. Similar to infrastructure ACL protection, layer-4 ACLs generally need to be applied as close to the edge of the network as possible, even though the intent is usually to protect the customer edge rather than the provider core. Application of layer-4 DDoS protection to a network edge is often automated using Flowspec [RFC5575].

For example, a web site which normally only handled traffic on TCP ports 80 and 443 could be subject to a volumetric DDoS attack using NTP and DNS packets with randomised source IP address, thereby rendering useless traditional [RFC5635] source-based real-time black hole mechanisms. In this situation, DDoS protection ACLs could be configured to block all UDP traffic at the network edge without impairing the web server functionality in any way. Thus, being able to block arbitrary protocols at the network edge can avoid DDoS-related problems both in the provider network and on the customer edge link.

4.2. Route-Processor Protection

Most modern routers have a fast hardware-assisted forwarding plane and a loosely coupled control plane, connected together with a link that has much less capacity than the forwarding plane could handle.

Traffic differentiation cannot be done by the control plane side, because this would overload the internal link connecting the forwarding plane to the control plane.

The Hop-by-Hop Options header is particularly challenging since, in most (if not all) implementations, it causes the corresponding packet to be punted to a software path. As a result, operators usually drop IPv6 packets containing this extension header. Please see [RFC6192] for advice regarding protection of the router control plane.

4.3. Inability to Perform Fine-grained Filtering

Some routers lack of fine-grained filtering of IPv6 extension headers. For example, an operator may want to drop packets containing Routing Header Type 0 (RHT0) but may only be able to filter on the extension header type (Routing Header). As a result, the operator may end up enforcing a more coarse filtering policy (e.g. "drop all packets containing a Routing Header" vs. "only drop packets that contain a Routing Header Type 0").

5. A Possible Attack Vector

The widespread drop of IPv6 packets employing IPv6 Extension Headers can, in some scenarios, be exploited for malicious purposes: if packets employing IPv6 EHs are known to be dropped on the path from system A to system B, an attacker could cause packets sent from A to B to be dropped by sending a forged ICMPv6 Packet Too Big (PTB) [RFC4443] error message to A (advertising an MTU smaller than 1280), such that subsequent packets from A to B include a fragment header (i.e., they result in atomic fragments [RFC6946]).

Possible scenarios where this attack vector could be exploited include (but are not limited to):

- Communication between any two systems through the public network (e.g., client from/to server or server from/to server), where packets with IPv6 extension headers are dropped by some intermediate router

- Communication between two BGP peers employing IPv6 transport, where these BGP peers implement ACLs to drop IPv6 fragments (to avoid control-plane attacks)

The aforementioned attack vector is exacerbated by the following factors:

- The attacker does not need to forge the IPv6 Source Address of his attack packets. Hence, deployment of simple BCP38 filters will not help as a counter-measure.

- Only the IPv6 addresses of the IPv6 packet embedded in the ICMPv6 payload need to be forged. While one could envision filtering devices enforcing BCP38-style filters on the ICMPv6 payload, the use of extension headers (by the attacker) could make this difficult, if not impossible.

- Many implementations fail to perform validation checks on the received ICMPv6 error messages, as recommended in Section 5.2 of [RFC4443] and documented in [RFC5927]. It should be noted that in some cases, such as when an ICMPv6 error message has (supposedly) been elicited by a connection-less transport protocol (or some other connection-less protocol being encapsulated in IPv6), it may be virtually impossible to perform validation checks on the received ICMPv6 error messages. And, because of IPv6 extension headers, the ICMPv6 payload might not even contain any useful information on which to perform validation checks.

- Upon receipt of one of the aforementioned ICMPv6 "Packet Too Big" error messages, the Destination Cache [RFC4861] is usually updated to reflect that any subsequent packets to such destination should include a Fragment Header. This means that a single ICMPv6 "Packet Too Big" error message might affect multiple communication instances (e.g. TCP connections) with such destination.

- A router or other middlebox cannot simply drop all incoming ICMPv6 Packet Too Big error messages, as this would create a PMTUD blackhole.

Possible mitigations for this issue include:

- Dropping incoming ICMPv6 Packet Too Big error messages that advertise an MTU smaller than 1280 bytes.

- Artificially reducing the MTU to 1280 bytes and dropping incoming ICMPv6 PTB error messages.

Both of these mitigations come at the expense of possibly preventing communication through SIIT [RFC6145], that relies on IPv6 atomic fragments (see [I-D.ietf-6man-deprecate-atomfrag-generation]), and also implies that the filtering device has the ability to filter ICMP PTB messages based on the contents of the MTU field.

[I-D.ietf-6man-deprecate-atomfrag-generation] documents while the generation of IPv6 atomic fragments is considered harmful, and

documents why this functionality is being removed from the upcoming revision of the core IPv6 protocol [I-D.ietf-6man-rfc2460bis]. Thus, any of the above mitigations would eliminate the attack vector without any interoperability implications.

6. Future Work

Based on the discussion provided in this document, we recommend the following (*non*-mutually exclusive) actions to improve the state of affairs of IPv6 extension headers:

- Vendors must allow for better granularity in the specification of filters for IPv6 extension headers, such that filters for specific EH types and subtypes (e.g. RHT0 vs. RHT2) can be specified wthout affecting other extension header types/subtypes unnecessarily (please see Section 4.3).

- Provide advice on the filtering of IPv6 packets that contain IPv6 extension headers (as in [I-D.ietf-opsec-ipv6-eh-filtering]).

- The IETF should evaluate the possibility of enforcing a cap on the maximum length of an IPv6 EH chain (e.g., as proposed in [I-D.wkumari-long-headers]). If not at the protocol specification level (i.e., "Standards Track"), such a cap could be recommended as operational advice of the form "IPv6 implementations are expected to support EH chains as long as the they fit in the Path-MTU for the corresponding packets (see [RFC7112]). However, given current technology constraints, we specifically note that all implementations MUST support EH chains of at least X bytes, and MUST be able to process such EH chains (where necessary), without negative performance impact".

We explicitly note that the authors of this document do not (in any way) suggest or propose to depracate IPv6 extension headers and that, on the contrary, they propose actions to improve their state of affairs.

7. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

8. Security Considerations

The security implications of IPv6 extension headers are discussed in Section 3. A specific attack vector that could leverage the widespread dropping of packets with IPv6 EHs (along with possible

countermeasures) is discussed in Section 5. This document does not introduce any new security issues.

9. Acknowledgements

The authors would like to thank (in alphabetical order) Mikael Abrahamsson, Fred Baker, Brian Carpenter, Lee Howard, Sander Steffann, Eric Vyncke, and Andrew Yourtchenko, for providing valuable comments on earlier versions of this document. Additionally, the authors would like to thank participants of the v6ops working group for their valuable input on the topics that led to the publication of this document.

Fernando Gont would like to thank Sander Steffann, who took the time to meet to discuss this document, even while higher priority events were in place.

Fernando Gont would like to thank Jan Zorz / Go6 Lab <http://go6lab.si/>, and Jared Mauch / NTT America, for providing access to systems and networks that were employed to perform experiments and measurements involving packets with IPv6 Extension Headers. Additionally, he would like to thank SixXS <https://www.sixxs.net> for providing IPv6 connectivity.

10. References

10.1. Normative References

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

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

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, DOI 10.17487/RFC6145, April 2011, <http://www.rfc-editor.org/info/rfc6145>.

[RFC6946] Gont, F., "Processing of IPv6 "Atomic" Fragments", RFC 6946, DOI 10.17487/RFC6946, May 2013, <http://www.rfc-editor.org/info/rfc6946>.

10.2. Informative References

[Atlasis2012]
Atlasis, A., "Attacking IPv6 Implementation Using Fragmentation", BlackHat Europe 2012. Amsterdam, Netherlands. March 14-16, 2012, <https://media.blackhat.com/bh-eu-12/Atlasis/bh-eu-12-Atlasis-Attacking_IPv6-Slides.pdf>.

[Atlasis2014]
Atlasis, A., "A Novel Way of Abusing IPv6 Extension Headers to Evade IPv6 Security Devices", May 2014, <http://www.insinuator.net/2014/05/a-novel-way-of-abusing-ipv6-extension-headers-to-evade-ipv6-security-devices/>.

[BH-EU-2014]
Atlasis, A., Rey, E., and R. Schaefer, "Evasion of High-End IDPS Devices at the IPv6 Era", BlackHat Europe 2014, 2014, <https://www.ernw.de/download/eu-14-Atlasis-Rey-Schaefer-briefings-Evasion-of-HighEnd-IPS-Devices-wp.pdf>.

[Bonica-NANOG58]
Bonica, R., "IPv6 Extension Headers in the Real World v2.0", NANOG 58. New Orleans, Louisiana, USA. June 3-5, 2013, <https://www.nanog.org/sites/default/files/mon.general.fragmentation.bonica.pdf>.

[Cisco-EH-Cons]
Cisco, , "IPv6 Extension Headers Review and Considerations", October 2006, <http://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.pdf>.

[Gont-Chown-IEPG89]
Gont, F. and T. Chown, "A Small Update on the Use of IPv6 Extension Headers", IEPG 89. London, UK. March 2, 2014, <http://www.iepg.org/2014-03-02-ietf89/fgont-iepg-ietf89-eh-update.pdf>.

[Gont-IEPG88]
Gont, F., "Fragmentation and Extension header Support in the IPv6 Internet", IEPG 88. Vancouver, BC, Canada. November 13, 2013, <http://www.iepg.org/2013-11-ietf88/fgont-iepg-ietf88-ipv6-frag-and-eh.pdf>.

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

[I-D.ietf-6man-predictable-fragment-id]
Gont, F., "Security Implications of Predictable Fragment Identification Values", draft-ietf-6man-predictable-fragment-id-10 (work in progress), October 2015.

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

[I-D.ietf-opsec-ipv6-eh-filtering]
Gont, F., LIU, S., and R. Bonica, "Recommendations on Filtering of IPv6 Packets Containing IPv6 Extension Headers", draft-ietf-opsec-ipv6-eh-filtering-00 (work in progress), March 2015.

[I-D.ietf-v6ops-ipv6-ehs-in-real-world]
Gont, F., Linkova, J., Chown, T., and S. LIU, "Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World", draft-ietf-v6ops-ipv6-ehs-in-real-world-02 (work in progress), December 2015.

[I-D.kampanakis-6man-ipv6-eh-parsing]
Kampanakis, P., "Implementation Guidelines for parsing IPv6 Extension Headers", draft-kampanakis-6man-ipv6-eh-parsing-01 (work in progress), August 2014.

[I-D.taylor-v6ops-fragdrop]
Jaeggli, J., Colitti, L., Kumari, W., Vyncke, E., Kaeo, M., and T. Taylor, "Why Operators Filter Fragments and What It Implies", draft-taylor-v6ops-fragdrop-02 (work in progress), December 2013.

[I-D.wkumari-long-headers]
Kumari, W., Jaeggli, J., Bonica, R., and J. Linkova, "Operational Issues Associated With Long IPv6 Header Chains", draft-wkumari-long-headers-03 (work in progress), June 2015.

[IEPG94-Scudder]
Petersen, B. and J. Scudder, "Modern Router Architecture for Protocol Designers", IEPG 94. Yokohama, Japan. November 1, 2015, <http://www.iepg.org/2015-11-01-ietf94/IEPG-RouterArchitecture-jgs.pdf>.

[Linkova-Gont-IEPG90]
Linkova, J. and F. Gont, "IPv6 Extension Headers in the Real World v2.0", IEPG 90. Toronto, ON, Canada. July 20, 2014, <http://www.iepg.org/2014-07-20-ietf90/iepg-ietf90-ipv6-ehs-in-the-real-world-v2.0.pdf>.

[PMTUD-Blackholes]
De Boer, M. and J. Bosma, "Discovering Path MTU black holes on the Internet using RIPE Atlas", July 2012, <http://www.nlnetlabs.nl/downloads/publications/pmtu-black-holes-msc-thesis.pdf>.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009, <http://www.rfc-editor.org/info/rfc5575>.

[RFC5635] Kumari, W. and D. McPherson, "Remote Triggered Black Hole Filtering with Unicast Reverse Path Forwarding (uRPF)", RFC 5635, DOI 10.17487/RFC5635, August 2009, <http://www.rfc-editor.org/info/rfc5635>.

[RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, DOI 10.17487/RFC5927, July 2010, <http://www.rfc-editor.org/info/rfc5927>.

[RFC6192] Dugal, D., Pignataro, C., and R. Dunn, "Protecting the Router Control Plane", RFC 6192, DOI 10.17487/RFC6192, March 2011, <http://www.rfc-editor.org/info/rfc6192>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <http://www.rfc-editor.org/info/rfc6437>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <http://www.rfc-editor.org/info/rfc6438>.

[RFC6980] Gont, F., "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery", RFC 6980, DOI 10.17487/RFC6980, August 2013, <http://www.rfc-editor.org/info/rfc6980>.

[RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <http://www.rfc-editor.org/info/rfc7045>.

[RFC7112] Gont, F., Manral, V., and R. Bonica, "Implications of Oversized IPv6 Header Chains", RFC 7112, DOI 10.17487/RFC7112, January 2014, <http://www.rfc-editor.org/info/rfc7112>.

[RFC7113] Gont, F., "Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)", RFC 7113, DOI 10.17487/RFC7113, February 2014, <http://www.rfc-editor.org/info/rfc7113>.

[RFC7123] Gont, F. and W. Liu, "Security Implications of IPv6 on IPv4 Networks", RFC 7123, DOI 10.17487/RFC7123, February 2014, <http://www.rfc-editor.org/info/rfc7123>.

[RIPE-Atlas] RIPE, , "RIPE Atlas", <https://atlas.ripe.net/>.

[Zack-FW-Benchmark] Zack, E., "Firewall Security Assessment and Benchmarking IPv6 Firewall Load Tests", IPv6 Hackers Meeting #1, Berlin, Germany. June 30, 2013, <http://www.ipv6hackers.org/meetings/ipv6-hackers-1/zack-ipv6hackers1-firewall-security-assessment-and-benchmarking.pdf>.

Authors' Addresses

Fernando Gont
SI6 Networks / UTN-FRH
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

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

Nick Hilliard
INEX
4027 Kingswood Road
Dublin 24
IE

Email: nick@inex.ie

Gert Doering
SpaceNet AG
Joseph-Dollinger-Bogen 14
Muenchen D-80807
Germany

Email: gert@space.net

Will (Shucheng) Liu
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: liushucheng@huawei.com

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net

INTERNET-DRAFT
Intended Status: Informational
Expires: September 14, 2017

Tom Herbert
Quantonium
Petr Lapukhov
Facebook
March 13, 2017

Identifier-locator addressing for IPv6
draft-herbert-nvo3-ila-04

Abstract

This specification describes identifier-locator addressing (ILA) for IPv6. Identifier-locator addressing differentiates between location and identity of a network node. Part of an address expresses the immutable identity of the node, and another part indicates the location of the node which can be dynamic. Identifier-locator addressing can be used to efficiently implement overlay networks for network virtualization as well as solutions for use cases in mobility.

Table of Contents

1 Introduction 4
1.1 Terminology 4
2 Architectural overview 6
2.1 Addressing 6
2.2 Network topology 6
2.3 Translations and mappings 7
2.4 ILA routing 8
3 Address formats 9
3.1 ILA address format 9
3.2 Locators 9
3.3 Identifiers 9
3.3.1 Checksum neutral-mapping format 10
3.3.2 Identifier types 10
3.3.2.1 Interface identifiers 10
3.3.2.2 Locally unique identifiers 11
3.3.2.3 Virtual networking identifiers for IPv4 11
3.3.2.4 Virtual networking identifiers for IPv6 unicast . . 12
3.3.2.5 Virtual networking identifiers for IPv6 multicast . 13
3.4 Standard identifier representation addresses 14
3.4.1 SIR for locally unique identifiers 15
3.4.2 SIR for virtual addresses 15
3.4.3 SIR domains 16
4 Operation 16
4.1 Identifier to locator mapping 16
4.2 Address translations 16
4.2.1 SIR to ILA address translation 16
4.2.2 ILA to SIR address translation 17
4.3 Virtual networking operation 17
4.3.1 Crossing virtual networks 18
4.3.2 IPv4/IPv6 protocol translation 18
4.4 Transport layer checksums 18
4.4.1 Checksum-neutral mapping 19
4.4.2 Sending an unmodified checksum 20
4.5 Address selection 20
4.6 Duplicate identifier detection 20

4.7 ICMP error handling 21
4.7.1 Handling ICMP errors by ILA capable hosts 21
4.7.2 Handling ICMP errors by non-ILA capable hosts 21
4.8 Multicast 22
5 Motivation for ILA 22
5.1 Use cases 22
5.1.1 Multi-tenant virtualization 22
5.1.2 Datacenter virtualization 23
5.1.3 Device mobility 23
5.2 Alternative methods 24
5.2.1 ILNP 24
5.2.2 Flow label as virtual network identifier 24
5.2.3 Extension headers 25
5.2.4 Encapsulation techniques 25
6 IANA Considerations 26
7 References 27
7.1 Normative References 27
7.2 Informative References 27
8 Acknowledgments 28
Appendix A: Communication scenarios 29
A.1 Terminology for scenario descriptions 29
A.2 Identifier objects 30
A.3 Reference network for scenarios 30
A.4 Scenario 1: Object to task 31
A.5 Scenario 2: Object to Internet 31
A.6 Scenario 3: Internet to object 31
A.7 Scenario 4: Tenant system to service 32
A.8 Scenario 5: Object to tenant system 32
A.9 Scenario 6: Tenant system to Internet 33
A.10 Scenario 7: Internet to tenant system 33
A.11 Scenario 8: IPv4 tenant system to object 33
A.12 Tenant to tenant system in the same virtual network 34
A.12.1 Scenario 9: TS to TS in the same VN using IPV6 34
A.12.2 Scenario 10: TS to TS in same VN using IPv4 34
A.13 Tenant system to tenant system in different virtual networks 34
A.13.1 Scenario 11: TS to TS in different VNs using IPV6 . . . 34
A.13.2 Scenario 12: TS to TS in different VNs using IPv4 . . . 35
A.13.3 Scenario 13: IPv4 TS to IPv6 TS in different VNs . . . 35
Appendix B: unique identifier generation 36
B.1 Globally unique identifiers method 36
B.2 Universally Unique Identifiers method 36
Appendix C: Datacenter task virtualization 37
C.1 Address per task 37
C.2 Job scheduling 37
C.3 Task migration 38
C.3.1 Address migration 38
C.3.2 Connection migration 39

1 Introduction

This specification describes the address formats, protocol operation, and communication scenarios of identifier-locator addressing (ILA). In identifier-locator addressing, an IPv6 address is split into a locator and an identifier component. The locator indicates the topological location in the network for a node, and the identifier indicates the node's identity which refers to the logical or virtual node in communications. Locators are routable within a network, but identifiers typically are not. An application addresses a peer destination by identifier. Identifiers are mapped to locators for transit in the network. The on-the-wire address is composed of a locator and an identifier: the locator is sufficient to route the packet to a physical host, and the identifier allows the receiving host to translate and forward the packet to the addressed application.

With identifier-locator addressing network virtualization and addressing for mobility can be implemented in an IPv6 network without any additional encapsulation headers. Packets sent with identifier-locator addresses look like plain unencapsulated packets (e.g. TCP/IP packets). This method is transparent to the network, so protocol specific mechanisms in network hardware work seamlessly. These mechanisms include hash calculation for ECMP, NIC large segment offload, checksum offload, etc.

Many of the concepts for ILA are adapted from Identifier-Locator Network Protocol (ILNP) ([RFC6740], [RFC6741]) which defines a protocol and operations model for identifier-locator addressing in IPv6.

Section 5 provides a motivation for ILA and comparison of ILA with alternative methods that achieve similar functionality.

1.1 Terminology

ILA	Identifier-locator addressing.
ILA router	A network node that performs ILA translation and forwarding of translated packets.
ILA host	An end host that is capable of performing ILA translations on transmit or receive.
ILA node	A network node capable of performing ILA translations. This can be an ILA router or ILA host.
Locator	A network prefix that routes to a physical host.

Locators provide the topological location of an addressed node. In ILA locators are a sixty-four bit prefixes.

Identifier A number that identifies an addressable node in the network independent of its location. ILA identifiers are sixty-four bit values.

ILA address
An IPv6 address composed of a locator (upper sixty-four bits) and an identifier (low order sixty-four bits).

SIR Standard identifier representation.

SIR prefix A sixty-four bit network prefix used to identify a SIR address.

SIR address
An IPv6 address composed of a SIR prefix (upper sixty-four bits) and an identifier (lower sixty-four bits). SIR addresses are visible to applications and provide a means to address nodes independent of their location.

SIR domain A unique identifier namespace defined by a SIR prefix. Each SIR prefix defines a SIR domain.

ILA translation
The process of translating the upper sixty-four bits of an IPv6 address. Translations may be from a SIR prefix to a locator or a locator to a SIR prefix.

Virtual address
An IPv6 or IPv4 address that resides in the address space of a virtual network. Such addresses may be translated to SIR addresses as an external representation of the address outside of the virtual network, or they may be translated to ILA addresses for transit over an underlay network.

Topological address
An address that refers to a non-virtual node in a network topology. These address physical hosts in a network.

2 Architectural overview

Identifier-locator addressing allows a data plane method to implement network virtualization without encapsulation and its related overheads. The service ILA provides is effectively layer 3 over layer 3 network virtualization (IPv4 or IPv6 over IPv6).

2.1 Addressing

ILA performs translations on IPv6 address. There are two types of addresses introduced for ILA: ILA addresses and SIR addresses.

ILA addresses are IPv6 addresses that are composed of a locator (upper sixty-four bits) and an identifier (low order sixty-four bits). The identifier serves as the logical addresses of a node, and the locator indicates the location of the node on the network.

A SIR address (standard identifier representation) is an IPv6 address that contains an identifier and an application visible SIR prefix. SIR addresses are visible to the application and can be used as connection endpoints. When a packet is sent to a SIR address, an ILA router or host overwrites the SIR prefix with a locator corresponding to the identifier. When a peer ILA node receives the packet, the locator is overwritten with the original SIR prefix before delivery to the application. In this manner applications only see SIR addresses, they do not have visibility into ILA addresses.

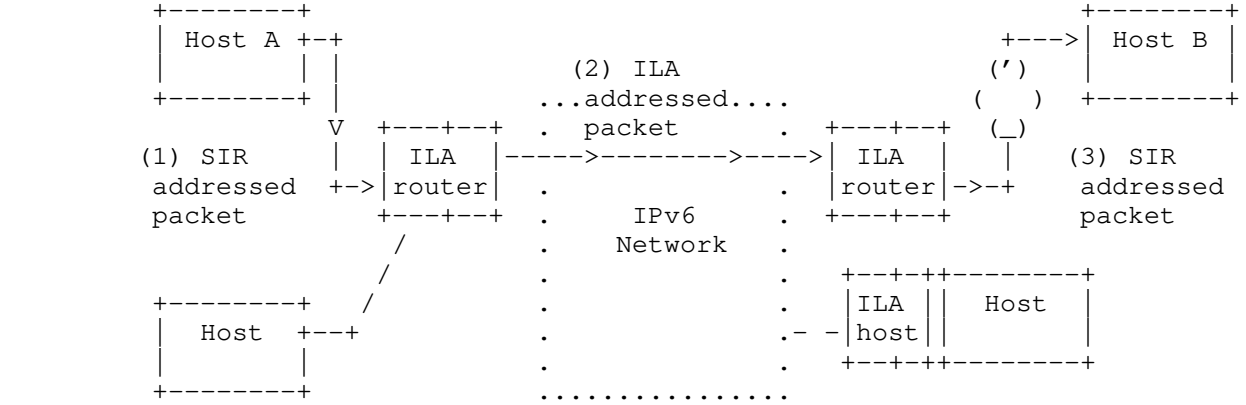
ILA translations can transform addresses from one type to another. In network virtualization virtual addresses can be translated into ILA and SIR addresses, and conversely ILA and SIR addresses can be translated to virtual addresses.

2.2 Network topology

ILA nodes are nodes in the network that perform ILA translations. An ILA router is a node that performs ILA address translation and packet forwarding to implement overlay network functionality. ILA routers perform translations on packets sent by end nodes for transport across an underlay network. Packets received by ILA routers on the underlay network have their addresses reversed translated for reception at an end node. An ILA host is an end node that implements ILA functionality for transmitting or receiving packets.

ILA nodes are responsible for transit of packets over an underlay network. On ingress to an ILA node (host or router) the virtual or SIR address of a destination is translated to an ILA address. At the a peer ILA node, the reverse translation is performed before handing packets to an application.

The figure below provides an example topology using ILA. ILA translations performed in one direction between Host A and Host B are denoted. Host A sends a packet with a destination SIR address (step (1)). An ILA router in the path translates the SIR address to an ILA address with a locator set to Host B, referring to the location of the node indicated by the identifier in the SIR address. The packet is forwarded over the network and delivered to a peer ILA node (step 2). The peer ILA node, in this case another ILA router, translates the destination address back to a SIR address and forwards to the final destination (step 3).



2.3 Translations and mappings

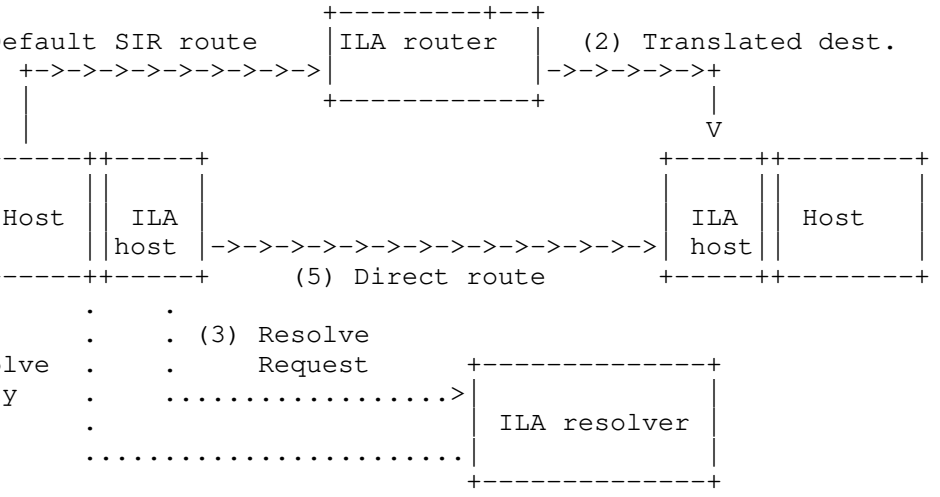
Address translation is the mechanism employed by ILA. Logical or virtual addresses are translated to topological IPv6 addresses for transport to the proper destination. Translation occurs in the upper sixty-four bits of an address, the low order sixty-four bits contains an identifier that is immutable and is not used to route a packet.

Each ILA node maintains a mapping table. This table maps identifiers to locators. The mappings are dynamic as nodes with identifiers can be created, destroyed, or migrated between physical hosts. Mappings are propagated amongst ILA routers or hosts in a network using mapping propagation protocols (mapping propagation protocols will be described in other specifications).

Identifiers are not statically bound to a host on the network, and in fact their binding (or location) may change. This is the basis for network virtualization and address migration. An identifier is mapped to a locator at any given time, and a set of identifier to locator mappings is propagated throughout a network to allow communications. The mappings are kept synchronized so that if an identifier migrates to a new physical host, its identifier to locator mapping is updated.

2.4 ILA routing

ILA is intended to be sufficiently lightweight so that all the hosts in a network could potentially send and receive ILA addressed packets. In order to scale this model and allow for hosts that do not participate in ILA, a routing topology may be applied. A simple routing topology is illustrated below.



An ILA router can be addressed by an "anycast" SIR prefix so that it receives packets sent on the network with SIR addresses. When an ILA router receives a SIR addressed packet (step (1) in the diagram) it will perform the ILA translation and send the ILA addressed packet to the destination ILA node (step (2)).

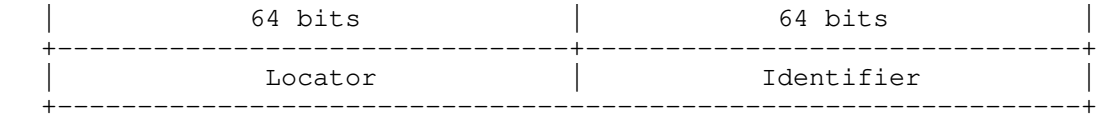
If a sending host is ILA capable the triangular routing can be eliminated by performing an ILA resolution protocol. This entails the host sending an ILA resolve request that specifies the SIR address to resolve (step (3) in the figure). An ILA resolver can respond to a resolver request with the identifier to locator mapping (step (4)). Subsequently, the ILA host can perform ILA translation and send directly to the destination specified in the locator (step (5) in the figure). The ILA resolution protocol will be specified in a companion document.

In this model an ILA host maintains a cache of identifier mappings for identifiers that it is currently communicating with. ILA routers are expected to maintain a complete list of identifier to locator mappings within the SIR domains that they service.

3 Address formats

3.1 ILA address format

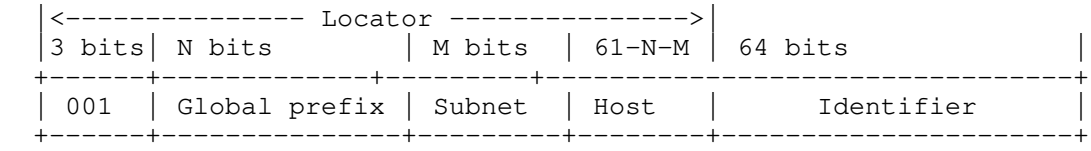
An ILA address is composed of a locator and an identifier where each occupies sixty-four bits (similar to the encoding in ILNP [RFC6741]).



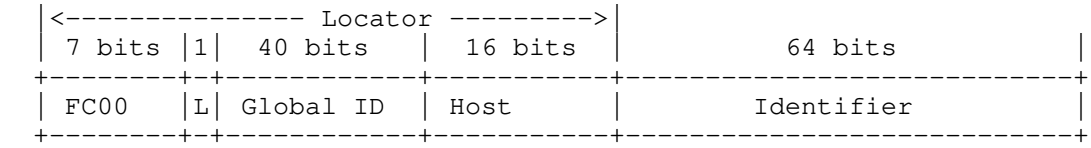
3.2 Locators

Locators are routable network address prefixes that create topological addresses for physical hosts within the network. They may be assigned from a global address block [RFC3587], or be based on unique local IPv6 unicast addresses as described in [RFC4193].

The format of an ILA address with a global unicast locator is:

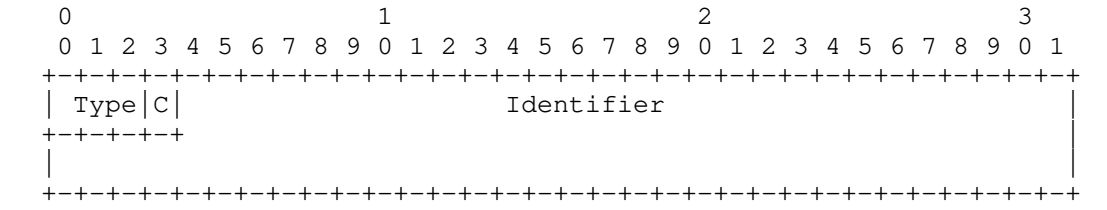


The format of an ILA address with a unique local IPv6 unicast locator is:



3.3 Identifiers

The format of an ILA identifier is:

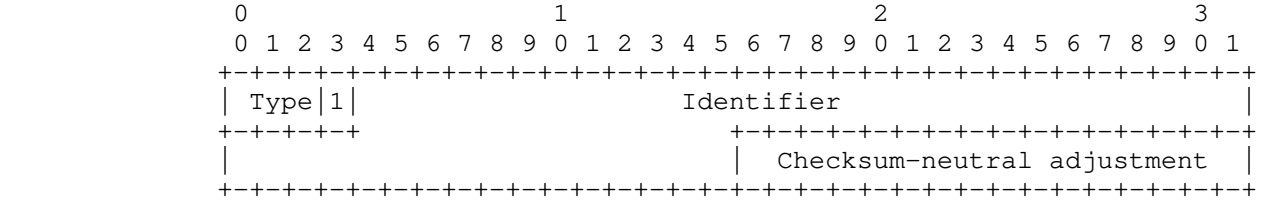


Fields are:

- Type: Type of the identifier (see section 3.3.2).
- C: The C-bit. This indicates that checksum-neutral mapping applied (see section 3.3.1).
- Identifier: Identifier value.

3.3.1 Checksum neutral-mapping format

If the C-bit is set the low order sixteen bits of an identifier contain the adjustment for checksum-neutral mapping (see section 4.4.1 for description of checksum-neutral mapping). The format of an identifier with checksum neutral mapping is:



3.3.2 Identifier types

Identifier types allow standard encodings for common uses of identifiers. Defined identifier types are:

0: interface identifier

1: locally unique identifier

2: virtual networking identifier for IPv4 address

3: virtual networking identifier for IPv6 unicast address

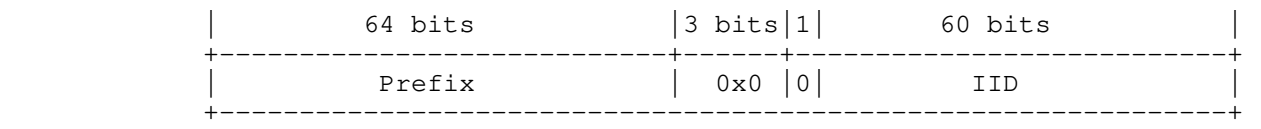
4: virtual networking identifier for IPv6 multicast address

5-7: Reserved

3.3.2.1 Interface identifiers

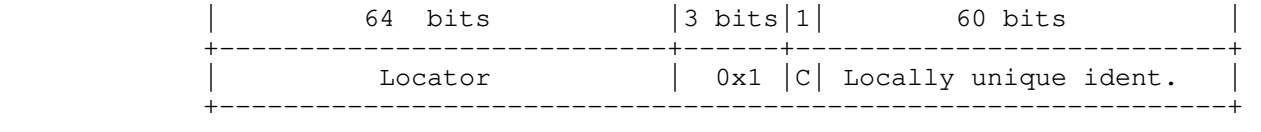
The interface identifier type indicates a plain local scope interface identifier. When this type is used the address is a normal IPv6 address without identifier-locator semantics. The purpose of this type is to allow normal IPv6 addresses to be defined within the same networking prefix as ILA addresses. Type bits and C-bit MUST be zero.

The format of an ILA interface identifier address is:

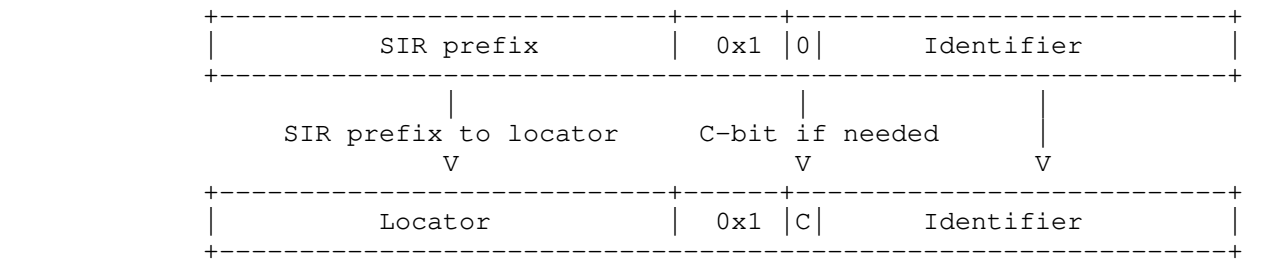


3.3.2.2 Locally unique identifiers

Locally unique identifiers (LUI) can be created for various addressable objects within a network. These identifiers are in a flat sixty bit space and must be unique within a SIR domain (unique within a site for instance). To simplify administration, hierarchical allocation of locally unique identifiers may be performed. The format of an ILA address with locally unique identifiers is:

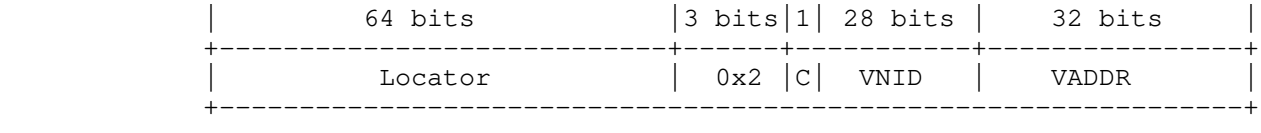


The figure below illustrates the translation from SIR address to an ILA address as would be performed when a node sends to a SIR address. Note the low order 16 bites of the identifier may be modified as the checksum-neutral adjustment. The reverse translation of ILA address to SIR address is symmetric.



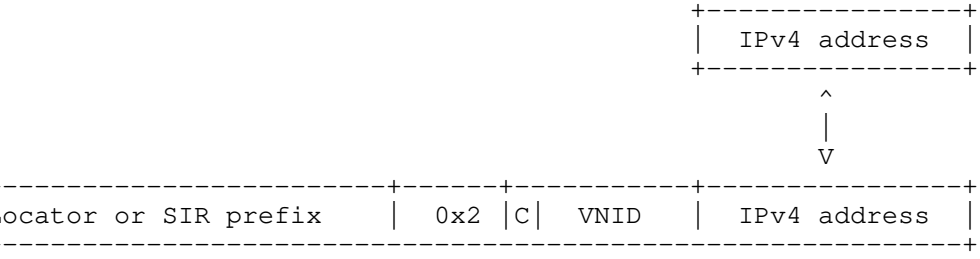
3.3.2.3 Virtual networking identifiers for IPv4

This type defines a format for encoding an IPv4 virtual address and virtual network identifier within an identifier. The format of an ILA address for IPv4 virtual networking is:



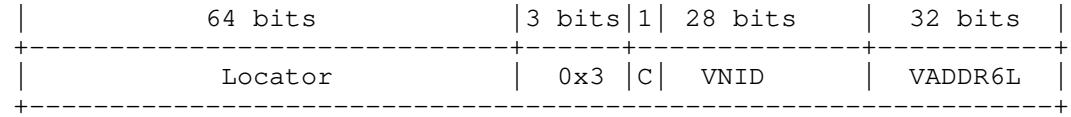
VNID is a virtual network identifier and VADDR is a virtual address within the virtual network indicated by the VNID. The VADDR can be an IPv4 unicast or multicast address, and may often be in a private address space (i.e. [RFC1918]) used in the virtual network.

Translating a virtual IPv4 address into an ILA or SIR address and the reverse translation are straight forward. Note that the low order 16 bits of the IPv6 address may be modified as the checksum-neutral adjustment and that this translation implies protocol translation when sending IPv4 packets over an ILA IPv6 network.



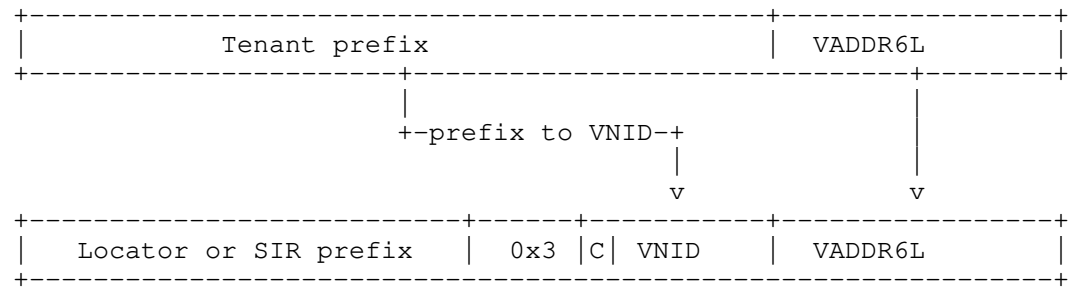
3.3.2.4 Virtual networking identifiers for IPv6 unicast

In this format, a virtual network identifier and virtual IPv6 unicast address are encoded within an identifier. To facilitate encoding of virtual addresses, there is a unique mapping between a VNID and a ninety-six bit prefix of the virtual address. The format an IPv6 unicast encoding with VNID in an ILA address is:

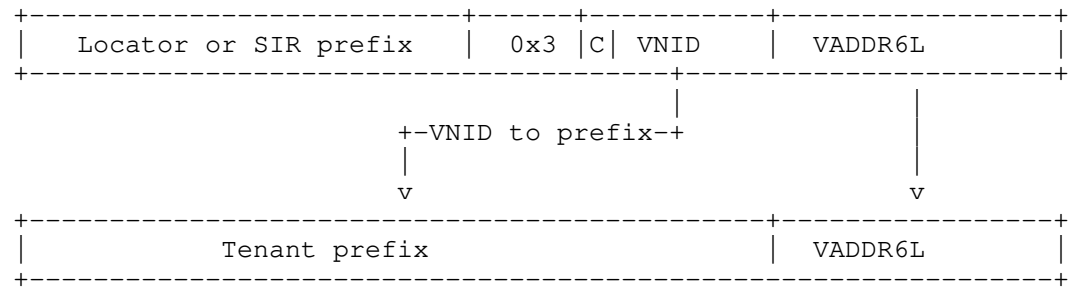


VADDR6L contains the low order 32 bits of the IPv6 virtual address. The upper 96 bits of the virtual address are inferred from the VNID to prefix mapping. Note that for ILA translations the low order sixteen of the VADDR6L may be modified for checksum-neutral adjustment.

The figure below illustrates encoding a tenant IPv6 virtual unicast address into a ILA or SIR address.

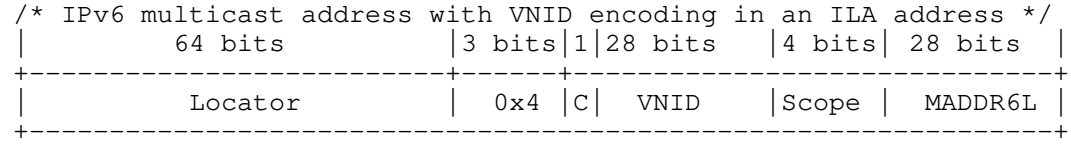


This encoding is reversible, given an ILA address, the virtual address visible to the tenant can be deduced:



3.3.2.5 Virtual networking identifiers for IPv6 multicast

In this format, a virtual network identifier and virtual IPv6 multicast address are encoded within an identifier.



This format encodes an IPv6 multicast address in an identifier. The scope indicates multicast address scope as defined in [RFC7346]. MADDR6L is the low order 28 bits of the multicast address. The full multicast address is thus:

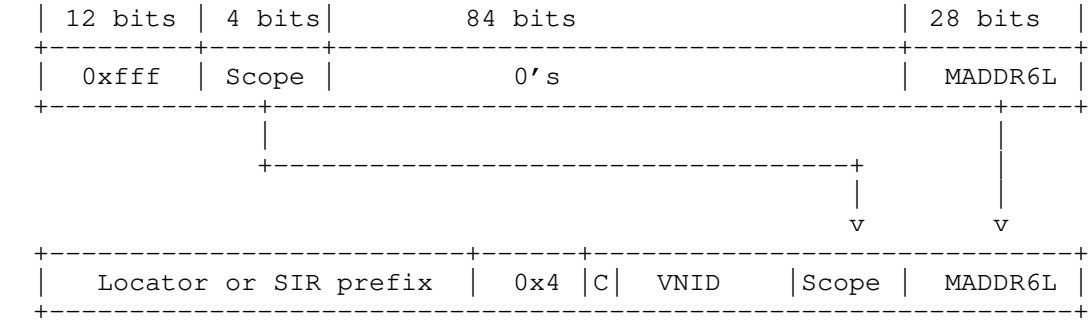
ff0<Scope>::<MADDRL6 high 12 bits>:<MADDRL6 low 16 bits>

And so can encode multicast addresses of the form:

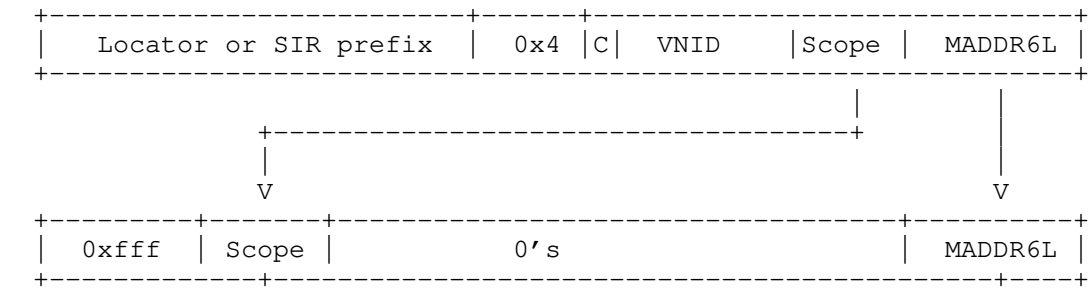
ff0X::0 to ff0X::0fff:ffff

The figure below illustrates encoding a tenant IPv6 virtual multicast

address in an ILA or SIR address. Note that low order sixteen bits of MADDR6L may be modified to be the checksum-neutral adjustment.



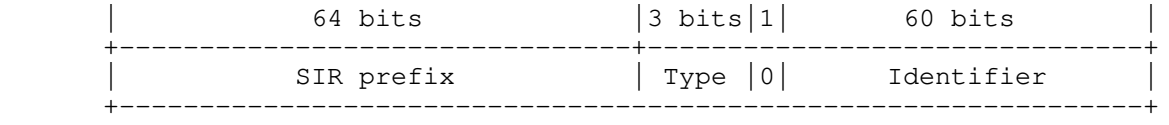
This translation is reversible:



3.4 Standard identifier representation addresses

An identifier identifies objects or nodes in a network. For instance, an identifier may refer to a specific host, virtual machine, or tenant system. When a host initiates a connection or sends a packet, it uses the identifier to indicate the peer endpoint of the communication. The endpoints of an established connection context also referenced by identifiers. It is only when the packet is actually being sent over a network that the locator for the identifier needs to be resolved.

In order to maintain compatibility with existing networking stacks and applications, identifiers are encoded in IPv6 addresses using a standard identifier representation (SIR) address. A SIR address is a combination of a prefix which occupies what would be the locator portion of an ILA address, and the identifier in its usual location. The format of a SIR address is:



The C-bit (checksum-neutral mapping) MUST be zero for a SIR address. Type may be any identifier type except zero (interface identifiers)

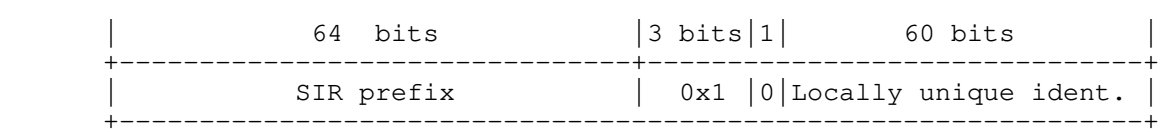
A SIR prefix may be site-local, or globally routable. A globally routable SIR prefix facilitates connectivity between hosts on the Internet and ILA nodes. A gateway between a site's network and the Internet can translate between SIR prefix and locator for an identifier. A network may have multiple SIR prefixes where each prefix defines a unique identifier space.

Locators MUST only be associated with one SIR prefix. This ensures that if a translation from a SIR address to an ILA address is performed when sending a packet, the reverse translation at the receiver yields the same SIR address that was seen at the transmitter. This also ensures that a reverse checksum-neutral mapping can be performed at a receiver to restore the addresses that were included in a pseudo header for setting a transport checksum.

A standard identifier representation address can be used as the externally visible address for a node. This can used throughout the network, returned in DNS AAAA records [RFC3363], used in logging, etc. An application can use a SIR address without knowledge that it encodes an identifier.

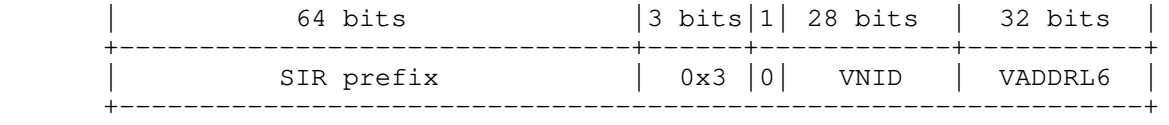
3.4.1 SIR for locally unique identifiers

The SIR address for a locally unique identifier has format:



3.4.2 SIR for virtual addresses

A virtual address can be encoded using the standard identifier representation. For example, the SIR address for an IPv6 virtual address may be:



Note that this allows three representations of the same address in he network: as a virtual address, a SIR address, and an ILA address.

3.4.3 SIR domains

Each SIR prefix defines a SIR domain. A SIR domain is a unique name space for identifiers within a domain. The full identity of a node is thus determined by an identifier and SIR domain (SIR prefix). Locators MUST map to only one SIR domain in order to ensure that translation from a locator to SIR prefix is unambiguous.

4 Operation

This section describes operation methods for using identifier-locator addressing.

4.1 Identifier to locator mapping

An application initiates a communication or flow using a SIR address or virtual address for a destination. In order to send a packet on the network, the destination address is translated by an ILA router or an ILA host in the path. An ILA node maintains a list of mappings from identifier to locator to perform this translation.

The mechanisms of propagating and maintaining identifier to locator mappings are outside the scope of this document.

4.2 Address translations

With ILA, address translation is performed to convert SIR addresses to ILA addresses, and ILA addresses to SIR addresses. Translation is usually done on a destination address as a form of source routing, however translation on source virtual addresses to SIR addresses can also be done to support some network virtualization scenarios (see appendix A.7 for example).

4.2.1 SIR to ILA address translation

When translating a SIR address to an ILA address the SIR prefix in the address is overridden with a locator, and checksum neutral mapping may be performed. Since this operation is potentially done for every packet the process should be very efficient (particularly the lookup and checksum processing operations).

The typical steps to transmit a packet using ILA are:

1) Host stack creates a packet with source address set to a local address (possibly a SIR address) for the local identity, and

the destination address is set to the SIR address or virtual address for the peer. The peer address may have been discovered through DNS or other means.

2) An ILA router or host translates the packet to use the locator. If the original destination address is a SIR address then the SIR prefix is overwritten with the locator. If the original packet is a virtually addressed tenant packet then the virtual address is translated per section 3.3.2. The locator is discovered by a lookup in the locator to identifier mappings.

3) The ILA node performs checksum-neutral mapping if configured for that (section 4.4.1).

4) Packet is forwarded on the wire. The network routes the packet to the host indicated by the locator.

4.2.2 ILA to SIR address translation

When a destination node (ILA router or end host) receives an ILA addressed packet, the ILA address MUST be translated back to a SIR address (or tenant address) before upper layer processing.

The steps of receive processing are:

1) Packet is received. The destination locator is verified to match a locator assigned to the host.

2) A lookup is performed on the destination identifier to find if it addresses a local identifier. If match is found, either the locator is overwritten with SIR prefix (for locally unique identifier type) or the address is translated back to a tenant virtual address as shown in appendix A.7.

3) Perform reverse checksum-neutral mapping if C-bit is set (section 4.4.1).

4) Perform any optional policy checks; for instance that the source may send a packet to the destination address, that packet is not illegitimately crossing virtual networks, etc.

5) Forward packet to application processing.

4.3 Virtual networking operation

When using ILA with virtual networking identifiers, address translation is performed to convert tenant virtual network and virtual addresses to ILA addresses, and ILA addresses back to a

virtual network and tenant's virtual addresses. Translation may occur on either source address, destination address, or both (see scenarios for virtual networking in Appendix A). Address translation is performed similar to the SIR translation cases described above.

4.3.1 Crossing virtual networks

With explicit configuration, virtual network hosts may communicate directly with virtual hosts in another virtual network by using SIR addresses for virtualization in both the source and destination addresses. This might be done to allow services in one virtual network to be accessed from another (by prior agreement between tenants). See appendix A.13 for example of ILA addressing for such a scenario.

4.3.2 IPv4/IPv6 protocol translation

An IPv4 tenant may send a packet that is converted to an IPv6 packet with ILA addresses. Similarly, an IPv6 packet with ILA addresses may be converted to an IPv4 packet to be received by an IPv4-only tenant. These are IPv4/IPv6 stateless protocol translations as described in [RFC6144] and [RFC6145]. See appendix A.12 for a description of these scenarios.

4.4 Transport layer checksums

Packets undergoing ILA translation may encapsulate transport layer checksums (e.g. TCP or UDP) that include a pseudo header that is affected by the translation.

ILA provides two alternatives do deal with this:

- Perform a checksum-neutral mapping to ensure that an encapsulated transport layer checksum is kept correct on the wire.

- Send the checksum as-is, that is send the checksum value based on the pseudo header before translation.

Some intermediate devices that are not the actual end point of a transport protocol may attempt to validate transport layer checksums. In particular, many Network Interface Cards (NICs) have offload capabilities to validate transport layer checksums (including any pseudo header) and return a result of validation to the host. Typically, these devices will not drop packets with bad checksums, they just pass a result to the host. Checksum offload is a performance benefit, so if packets have incorrect checksums on the wire this benefit is lost. With this incentive, applying a checksum-

neutral mapping is the recommended alternative. If it is known that the addresses of a packet are not included in a transport checksum, for instance a GRE packet is being encapsulated, then a source may choose not to perform checksum-neutral mapping.

4.4.1 Checksum-neutral mapping

When a change is made to one of the IP header fields in the IPv6 pseudo-header checksum (such as one of the IP addresses), the checksum field in the transport layer header may become invalid. Fortunately, an incremental change in the area covered by the Internet standard checksum [RFC1071] will result in a well-defined change to the checksum value [RFC1624]. So, a checksum change caused by modifying part of the area covered by the checksum can be corrected by making a complementary change to a different 16-bit field covered by the same checksum.

ILA can perform a checksum-neutral mapping when a SIR prefix or virtual address is translated to a locator in an IPv6 address, and performs the reverse mapping when translating a locator back to a SIR prefix or virtual address. The low order sixteen bits of the identifier contain the checksum adjustment value for ILA.

On transmission, the translation process is:

1) Compute the one's complement difference between the SIR prefix and the locator. Fold this value to 16 bits (add-with-carry four 16-bit words of the difference).

2) Add-with-carry the bit-wise not of the 0x1000 (i.e. 0xefff) to the value from #1. This compensates the checksum for setting the C-bit.

3) Add-with-carry the value from #2 to the low order sixteen bits of the identifier.

4) Set the resultant value from #3 in the low order sixteen bits of the identifier and set the C-bit.

Note that the "adjustment" (the 16-bit value set in the identifier in set #3) is fixed for a given SIR to locator mapping, so the adjustment value can be saved in an associated data structure for a mapping to avoid computing it for each translation.

On reception of an ILA addressed packet, if the C-bit is set in an ILA address:

1) Compute the one's complement difference between the locator in

the address and the SIR prefix that the locator is being translated to. Fold this value to 16 bits (add-with-carry four 16-bit words of the difference).

2) Add-with-carry 0x1000 to the value from #1. This compensates the checksum for clearing the C-bit.

3) Add-with-carry the value from #2 to the low order sixteen bits of the identifier.

4) Set the resultant value from #3 in the low order sixteen bits of the identifier and clear the C-bit. This restores the original identifier sent in the packet.

4.4.2 Sending an unmodified checksum

When sending an unmodified checksum, the checksum is incorrect as viewed in the packet on the wire. At the receiver, ILA translation of the destination ILA address back to the SIR address occurs before transport layer processing. This ensures that the checksum can be verified when processing the transport layer header containing the checksum. Intermediate devices are not expected to drop packets due to a bad transport layer checksum.

4.5 Address selection

There may be multiple possibilities for creating either a source or destination address. A node may be associated with more than one identifier, and there may be multiple locators for a particular identifier. The choice of locator or identifier is implementation or configuration specific. The selection of an identifier occurs at flow creation and must be invariant for the duration of the flow. Locator selection must be done at least once per flow, and the locator associated with the destination of a flow may change during the lifetime of the flow (for instance in the case of a migrating connection it will change). ILA address selection should follow specifications in Default Address Selection for Internet Protocol Version 6 (IPv6) [RFC6724].

4.6 Duplicate identifier detection

As part of implementing the locator to identifier mapping, duplicate identifier detection should be implemented in a centralized control plane. A registry of identifiers could be maintained (possibly in association the identifier to locator mapping database). When a node creates an identifier it registers the identifier, and when the identifier is no longer in use (e.g. task completes) the identifier is unregistered. The control plane should able to detect a

registration attempt for an existing identifier and deny the request.

4.7 ICMP error handling

A packet that contains an ILA address may cause ICMP errors within the network. In this case the ICMP data contains an IP header with an ILA address. ICMP messages are sent back to the source address in the packet. Upon receiving an ICMP error the host will process it differently depending on whether it is ILA capable.

4.7.1 Handling ICMP errors by ILA capable hosts

If a host is ILA capable it can attempt to reverse translate the ILA address in the destination of a header in the ICMP data back to a SIR address that was originally used to transmit the packet. The steps are:

1) Assume that the upper sixty-four bits of the destination address in the ICMP data is a locator. Try match these bits back to a SIR address. If the host is only in one SIR domain, then the mapping to SIR address is implicit. If the host is in multiple domains then a locator to SIR addresses table can be maintained for this lookup.

2) If the identifier is marked with checksum-neutral mapping, undo the checksum-neutral using the SIR address found in #1. The resulting identifier address is potentially the original address used to send the packet.

3) Lookup the identifier in the identifier to locator mapping table. If an entry is found compare the locator in the entry to the locator (upper sixty-four bits) of the destination address in the IP header of the ICMP data. If these match then proceed to next step.

4) Overwrite the upper sixty-four bits of the destination address in the ICMP data with the found SIR address and overwrite the low order sixty-four bits with the found identifier (the result of undoing checksum-neutral mapping). The resulting address should be the original SIR address used in sending. The ICMP error packet can then be received by the stack for further processing.

4.7.2 Handling ICMP errors by non-ILA capable hosts

A non-ILA capable host may receive an ICMP error generated by the network that contains an ILA address in an IP header contained in the ICMP data. This would happen in the case that an ILA router performed

translation on a packet the host sent and that packet subsequently generated an ICMP error. In this case the host receiving the error message will attempt to find the connection state corresponding to the packet in headers the ICMP data. Since the host is unaware of ILA the lookup for connection state should fail. Because the host cannot recover the original addresses it used to send the packet, it won't be able any to derive any useful information about the original destination of the packet that it sent.

If packets for a flow are always routed through an ILA router in both directions, for example ILA routers are coincident with edge routes in a network, then ICMP errors could be intercepted by an intermediate node which could translate the destination addresses in ICMP data back to the original SIR addresses. A receiving host would then see the destination address in the packet of the ICMP data to be that it used to transmit the original packet.

4.8 Multicast

ILA is generally not intended for use with multicast. In the case of multicast, routing of packets is based on the source address. Neither the SIR address nor an ILA address is suitable for use as a source address in a multicast packet. A SIR address is unroutable and hence would make a multicast packet unroutable if used as a source address. Using an ILA address as the source address makes the multicast packet routable, but this exposes ILA address to applications which is especially problematic on a multicast receiver that doesn't support ILA.

If all multicast receivers are known to support ILA, a local locator address may be used in the source address of the multicast packet. In this case, each receiver will translate the source address from an ILA address to a SIR address before delivering packets to an application.

5 Motivation for ILA

5.1 Use cases

5.1.1 Multi-tenant virtualization

In multi-tenant virtualization overlay networks are established for tenants to provide virtual networks. Each tenant may have one or more virtual networks and a tenant's nodes are assigned virtual addresses within virtual networks. Identifier-locator addressing may be used as an alternative to traditional network virtualization encapsulation protocols used to create overlay networks (e.g. VXLAN [RFC7348]). Section 5.2.4 describes the advantages of using ILA in lieu of

encapsulation protocols.

Tenant systems (e.g. VMs) run on physical hosts and may migrate to different hosts. A tenant system is identified by a virtual address and virtual networking identifier of a corresponding virtual network. ILA can encode the virtual address and a virtual networking identifier in an ILA identifier. Each identifier is mapped to a locator that indicates the current host where the tenant system resides. Nodes that send to the tenant system set the locator per the mapping. When a tenant system migrates its identifier to locator mapping is updated and communicating nodes will use the new mapping.

5.1.2 Datacenter virtualization

Datacenter virtualization virtualizes networking resources. Various objects within a datacenter can be assigned addresses and serve as logical endpoints of communication. A large address space, for example that of IPv6, allows addressing to be used beyond the traditional concepts of host based addressing. Addressed objects can include tasks, virtual IP addresses (VIPs), pieces of content, disk blocks, etc. Each object has a location which is given by the host on which an object resides. Some objects may be migratable between hosts such that their location changes over time.

Objects are identified by a unique identifier within a namespace for the datacenter (appendix B discusses methods to create unique identifiers for ILA). Each identifier is mapped to a locator that indicates the current host where the object resides. Nodes that send to an object set the locator per the mapping. When an object migrates its identifier to locator mapping is updated and communicating nodes will use the new mapping.

A datacenter object of particular interest is tasks, units of execution for for applications. The goal of virtualzing tasks is to maximize resource efficiency and job scheduling. Tasks share many properties of tenant systems, however they are finer grained objects, may have a shorter lifetimes, and are likely created in greater numbers. Appendix C provides more detail and motivation for virtualizing tasks using ILA.

5.1.3 Device mobility

ILA may be applied as a solution for mobile devices. These are devices, smart phones for instance, that physically move between different networks. The goal of mobility is to provide a seamless transition when a device moves from one network to another.

Each mobile device is identified by unique identifier within some

provider domain. ILA encodes the identifier for the device in an ILA identifier. Each identifier is mapped to a locator that indicates the current network or point of attachment for the device. Nodes that send to the device set the locator per the mapping. When a mobile device moves between networks its identifier to locator mapping is updated and communicating nodes will use the new mapping.

5.2 Alternative methods

This section discusses the merits of alternative solution that have been proposed to provide network virtualization or mobility in IPv6.

5.2.1 ILNP

ILNP splits an address into a locator and identifier in the same manner as ILA. ILNP has characteristics, not present in ILA, that prevent it from being a practical solution:

- ILNP requires that transport layer protocol implementations must be modified to work over ILNP.
- ILNP can only be implemented in end hosts, not within the network. This essentially requires that all end hosts need to be modified to participate in mobility.
- ILNP employs IPv6 extension headers which are mostly considered non-deployable. ILA does not use these.
- Core support for ILA is in upstream Linux, to date there is no publicly available source code for ILNP.
- ILNP involves DNS to distribute mapping information, ILA assumes mapping information is not part of naming.

5.2.2 Flow label as virtual network identifier

The IPv6 flow label could conceptually be used as a 20-bit virtual network identifier in order to indicate a packet is sent on an overlay network. In this model the addresses may be virtual addresses within the specified virtual network. Presumably, the tuple of flow-label and addresses could be used by switches to forward virtually addressed packets.

This approach has some issues:

- Forwarding virtual packets to their physical location would require specialized switch support.

- The flow label is only twenty bits, this is too small to be a discriminator in forwarding a virtual packet to a specific destination. Conceptually, the flow label might be used in a type of label switching to solve that.

- The flow label is not considered immutable in transit, intermediate devices may change it.

- The flow label is not part of the pseudo header for transport checksum calculation, so it is not covered by any transport (or other) checksums.

5.2.3 Extension headers

To accomplish network virtualization an extension header, as a destination or routing option, could be used that contains the virtual destination address of a packet. The destination address in the IPv6 header would be the topological address for the location of the virtual node. Conceivably, segment routing could be used to implement network virtualization in this manner.

This technique has some issues:

- Intermediate devices must not insert extension headers [RFC2460bis].

- Extension headers introduce additional packet overhead which may impact performance.

- Extension headers are not covered by transport checksums (as the addresses would be) nor any other checksum.

- Extension headers are not widely supported in network hardware or devices. For instance, several NIC offloads don't work in the presence of extension headers.

5.2.4 Encapsulation techniques

Various encapsulation techniques have been proposed for implementing network virtualization and mobility. LISP is an example of an encapsulation that is based on locator identifier separation similar to ILA. The primary drawback of encapsulation is complexity and per packet overhead. For, instance when LISP is used with IPv6 the encapsulation overhead is fifty-six bytes and two IP headers are present in every packet. This adds considerable processing costs, requires considerations to handle path MTU correctly, and certain network accelerations may be lost.

6 IANA Considerations

There are no IANA considerations in this specification.

7 References

7.1 Normative References

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

[RFC2460bis] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", draft-ietf-6man-rfc2460bis-03, January 2016.

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

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, June 2011.

[RFC1071] Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum", RFC 1071, September 1988.

[RFC1624] Rijsinghani, A., "Computation of the Internet Checksum via Incremental Update", RFC 1624, May 1994.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

7.2 Informative References

[RFC6740] RJ Atkinson and SN Bhatti, "Identifier-Locator Network Protocol (ILNP) Architectural Description", RFC 6740, November 2012.

[RFC6741] RJ Atkinson and SN Bhatti, "Identifier-Locator Network Protocol (ILNP) Engineering Considerations", RFC 6741, November 2012.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC3363] Bush, R., Durand, A., Fink, B., Gudmundsson, O., and T. Hain, "Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)", RFC 3363, August 2002.

[RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global

Unicast Address Format", RFC 3587, August 2003.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

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

[NVO3ARCH] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and Narten, T., "An Architecture for Overlay Networks (NVO3)", draft-ietf-nvo3-arch-03

[GUE] Herbert, T., and Yong, L., "Generic UDP Encapsulation", draft-herbert-gue-02, work in progress.

[GUESEC] Yong, L., and Herbert, T. "Generic UDP Encapsulation (GUE) for Secure Transport", draft-hy-gue-4-secure-transport-00, work in progress

8 Acknowledgments

The author would like to thank Mark Smith, Lucy Yong, Erik Kline, Saleem Bhatti, Petr Lapukhov, Blake Matheny, Doug Porter, Pierre Pfister, and Fred Baker for their insightful comments for this draft; Roy Bryant, Lorenzo Colitti, Mahesh Bandewar, and Erik Kline for their work on defining and applying ILA.

Appendix A: Communication scenarios

This section describes the use of identifier-locator addressing in several scenarios.

A.1 Terminology for scenario descriptions

A formal notation for identifier-locator addressing with ILNP is described in [RFC6740]. We extend this to include for network virtualization cases.

Basic terms are:

A = IP Address
I = Identifier
L = Locator
LUI = Locally unique identifier
VNI = Virtual network identifier
VA = An IPv4 or IPv6 virtual address
VAX = An IPv6 networking identifier (IPv6 VA mapped to VAX)
SIR = Prefix for standard identifier representation
VNET = IPv6 prefix for a tenant (assumed to be globally routable)
Iaddr = IPv6 address of an Internet host

An ILA IPv6 address is denoted by

L:I

A SIR address with a locally unique identifier and SIR prefix is denoted by

SIR:LUI

A virtual identifier with a virtual network identifier and a virtual IPv4 address is denoted by

VNI:VA

An ILA IPv6 address with a virtual networking identifier for IPv4 would then be denoted

L:(VNI:VA)

The local and remote address pair in a packet or endpoint is denoted

A,A

An address translation sequence from SIR addresses to ILA addresses

for transmission on the network and back to SIR addresses at a receiver has notation:

A,A -> L:I,A -> A,A

A.2 Identifier objects

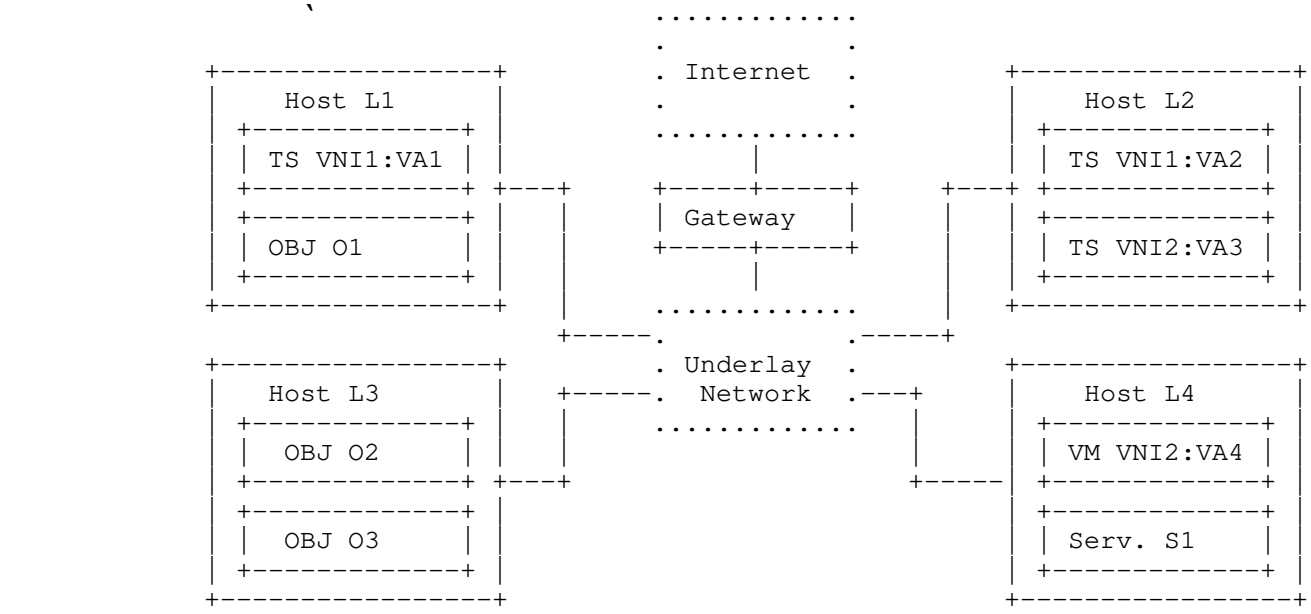
Identifier-locator addressing is broad enough in scope to address many different types of networking entities. For the purposes of this section we classify these as "objects" and "tenant systems".

Objects encompass uses where nodes are address by local unique identifiers (LUI). In the scenarios below objects are denoted by OBJ.

Tenant systems are those associated with network virtualization that have virtual addresses (that is they are addressed by VNI:VA). In the scenarios below tenant systems are denoted by TS.

A.3 Reference network for scenarios

The figure below provides an example network topology with ILA addressing in use. In this example, there are four hosts in the network with locators L1, L2, L3, and L4. There three objects with identifiers O1, O2, and O3, as well as a common networking service with identifier S1. There are two virtual networks VNI1 and VNI2, and four tenant systems addressed as: VA1 and VA2 in VNI1, VA3 and VA4 in VNI2. The network is connected to the Internet via a gateway.



Several communication scenarios can be considered:

1) Object to object
2) Object to Internet
3) Internet to object
4) Tenant system to local service
5) Object to tenant system
6) Tenant system to Internet
7) Internet to tenant system
8) IPv4 tenant system to service
9) Tenant system to tenant system same virtual network using IPv6
10) Tenant system to tenant system in same virtual network using IPv4
11) Tenant system to tenant system in different virtual network using IPv6
12) Tenant system to tenant system in different virtual network using IPv4
13) IPv4 tenant system to IPv6 tenant system in different virtual networks

A.4 Scenario 1: Object to task

The transport endpoints for object to object communication are the SIR addresses for the objects. When a packet is sent on the wire, the locator is set in the destination address of the packet. On reception the destination addresses is converted back to SIR representation for processing at the transport layer.

If object O1 is communicating with object O2, the ILA translation sequence would be:

```
SIR:O1,SIR:O2 ->                    // Transport endpoints on O1
SIR:O1,L3:O2 ->                     // ILA used on the wire
SIR:O1,SIR:O2                       // Received at O2
```

A.5 Scenario 2: Object to Internet

Communication from an object to the Internet is accomplished through use of a SIR address (globally routable) in the source address of packets. No ILA translation is needed in this path.

If object O1 is sending to an address Iaddr on the Internet, the packet addresses would be:

```
SIR:O1,Iaddr
```

A.6 Scenario 3: Internet to object

An Internet host transmits a packet to a task using an externally routable SIR address. The SIR prefix routes the packet to a gateway for the datacenter. The gateway translates the destination to an ILA address.

If a host on the Internet with address Iaddr sends a packet to object O3, the ILA translation sequence would be:

```
  Iaddr,SIR:O3 ->                    // Transport endpoint at Iaddr
  Iaddr,L1:O3 ->                     // On the wire in datacenter
  Iaddr,SIR:O3                       // Received at O3
```

A.7 Scenario 4: Tenant system to service

A tenant can communicate with a datacenter service using the SIR address of the service.

If TS VA1 is communicating with service S1, the ILA translation sequence would be:

```
  VNET:VA1,Saddr->                   // Transport endpoints in TS
  SIR:(VNET:VA1):Saddr->             // On the wire
  SIR:(VNET:VA1):Saddr               // Received at S1
```

Where VNET is the address prefix for the tenant and Saddr is the IPv6 address of the service.

The ILA translation sequence in the reverse path, service to tenant system, would be:

```
  Saddr,SIR:(VNET:VA1)               // Transport endpoints in S1
  Saddr,L1:(VNET:VA1)                // On the wire
  Saddr,VNET:VA1                     // Received at the TS
```

Note that from the point of view of the service task there is no material difference between a peer that is a tenant system versus one which is another task.

A.8 Scenario 5: Object to tenant system

An object can communicate with a tenant system through it’s externally visible address.

If object O2 is communicating with TS VA4, the ILA translation sequence would be:

```
  SIR:O2,VNET:VA4 ->                 // Transport endpoints at T2
  SIR:O2,L4:(VNI2:VAX4) ->           // On the wire
```

```
SIR:O2,VNET:VA4                    // Received at TS
```

A.9 Scenario 6: Tenant system to Internet

Communication from a TS to the Internet assumes that the VNET for the TS is globally routable, hence no ILA translation would be needed.

If TS VA4 sends a packet to the Internet, the addresses would be:

```
VNET:VA4,Iaddr
```

A.10 Scenario 7: Internet to tenant system

An Internet host transmits a packet to a tenant system using an externally routable tenant prefix and address. The prefix routes the packet to a gateway for the datacenter. The gateway translates the destination to an ILA address.

If a host on the Internet with address Iaddr is sending to TS VA4, the ILA translation sequence would be:

```
Iaddr,VNET:VA4 ->                  // Endpoint at Iaddr
Iaddr,L4:(VNI2:VAX4) ->            // On the wire in datacenter
Iaddr,VNET:VA4                     // Received at TS
```

A.11 Scenario 8: IPv4 tenant system to object

A TS that is IPv4-only may communicate with an object using protocol translation. The object would be represented as an IPv4 address in the tenant's address space, and stateless NAT64 should be usable as described in [RFC6145].

If TS VA2 communicates with object O3, the ILA translation sequence would be:

```
VA2,ADDR3 ->                       // IPv4 endpoints at TS
SIR:(VNI1:VA2),L3:O3 ->            // On the wire in datacenter
SIR:(VNI1:VA2),SIR:O3              // Received at task
```

VA2 is the IPv4 address in the tenant's virtual network, ADDR4 is an address in the tenant's address space that maps to the network service.

The reverse path, task sending to a TS with an IPv4 address, requires a similar protocol translation.

For object O3 communicate with TS VA2, the ILA translation sequence would be:

```
SIR:O3,SIR:(VNI1:VA2) ->          // Endpoints at T4
SIR:O3,L2:(VNI1:VA2) ->           // On the wire in datacenter
ADDR4,VA2                         // IPv4 endpoint at TS
```

A.12 Tenant to tenant system in the same virtual network

ILA may be used to allow tenants within a virtual network to communicate without the need for explicit encapsulation headers.

A.12.1 Scenario 9: TS to TS in the same VN using IPV6

If TS VA1 sends a packet to TS VA2, the ILA translation sequence would be:

```
VNET:VA1,VNET:VA2 ->              // Endpoints at VA1
VNET:VA1,L2:(VNI1,VAX2) ->        // On the wire
VNET:VA1,VNET:VA2 ->              // Received at VA2
```

A.12.2 Scenario 10: TS to TS in same VN using IPv4

For two tenant systems to communicate using IPv4 and ILA, IPv4/IPv6 protocol translation is done both on the transmit and receive.

If TS VA1 sends an IPv4 packet to TS VA2, the ILA translation sequence would be:

```
VA1,VA2 ->                        // Endpoints at VA1
SIR:(VNI1:VA1),L2:(VNI1,VA2) ->   // On the wire
VA1,VA2                           // Received at VA2
```

Note that the SIR is chosen by an ILA node as an appropriate SIR prefix in the underlay network. Tenant systems do not use SIR address for this communication, they only use virtual addresses.

A.13 Tenant system to tenant system in different virtual networks

A tenant system may be allowed to communicate with another tenant system in a different virtual network. This should only be allowed with explicit policy configuration.

A.13.1 Scenario 11: TS to TS in different VNs using IPV6

For TS VA4 to communicate with TS VA1 using IPv6 the translation sequence would be:

```
VNET2:VA4,VNET1:VA1->             // Endpoint at VA4
VNET2:VA4,L1:(VNI1,VAX1)->        // On the wire
VNET2:VA4,VNET1:VA1               // Received at VA1
```

Note that this assumes that VNET1 and VNET2 are globally routable between the two virtual networks.

A.13.2 Scenario 12: TS to TS in different VNs using IPv4

To allow IPv4 tenant systems in different virtual networks to communicate with each other, an address representing the peer would be mapped into each tenant's address space. IPv4/IPv6 protocol translation is done on transmit and receive.

For TS VA4 to communicate with TS VA1 using IPv4 the translation sequence may be:

```
VA4,SADDR1 ->                          // IPv4 endpoint at VA4
SIR:(VNI2:VA4),L1:(VNI1,VA1)->         // On the wire
SADDR4,VA1                             // Received at VA1
```

SADDR1 is the mapped address for VA1 in VA4's address space, and SADDR4 is the mapped address for VA4 in VA1's address space.

A.13.3 Scenario 13: IPv4 TS to IPv6 TS in different VNs

Communication may also be mixed so that an IPv4 tenant system can communicate with an IPv6 tenant system in another virtual network. IPv4/IPv6 protocol translation is done on transmit.

For TS VA4 using IPv4 to communicate with TS VA1 using IPv6 the translation sequence may be:

```
VA4,SADDR1 ->                          // IPv4 endpoint at VA4
SIR:(VNI2:VA4),L1:(VNI1,VAX1)->        // On the wire
SIR:(VNI2:VA4),VNET1:VA1               // Received at VA1
```

SADDR1 is the mapped IPv4 address for VA1 in VA4's address space.

In the reverse direction, TS VA1 using IPv6 would communicate with TS VA4 with the translation sequence:

```
VNET1:VA1,SIR:(VNI2:VA4)               // Endpoint at VA1
VNET1:VA1,L4:(VNI2:VA4)                // On the wire
SADDR1,VA4                             // Received at VA4
```

Appendix B: unique identifier generation

The unique identifier type of ILA identifiers can address 2**60 objects. This appendix describes some method to perform allocation of identifiers for objects to avoid duplicated identifiers being allocated.

B.1 Globally unique identifiers method

For small to moderate sized deployments the technique for creating locally assigned global identifiers described in [RFC4193] could be used. In this technique a SHA-1 digest of the time of day in NTP format and an EUI-64 identifier of the local host is performed. N bits of the result are used as the globally unique identifier.

The probability that two or more of these IDs will collide can be approximated using the formula:

```
P = 1 - exp(-N**2 / 2**(L+1))
```

where P is the probability of collision, N is the number of identifiers, and L is the length of an identifier.

The following table shows the probability of a collision for a range of identifiers using a 60-bit length.

Identifiers	Probability of Collision
1000	4.3368*10^-13
10000	4.3368*10^-11
100000	4.3368*10^-09
1000000	4.3368*10^-07

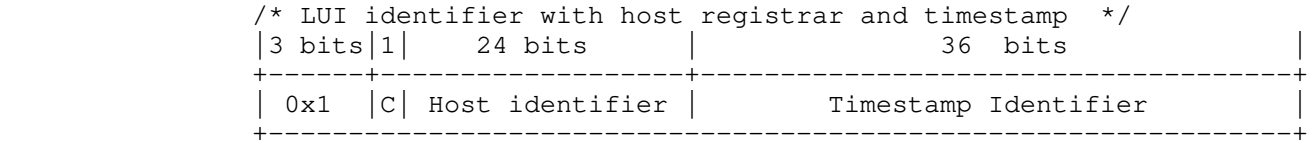
Note that locally unique identifiers may be ephemeral, for instance a task may only exist for a few seconds. This should be considered when determining the probability of identifier collision.

B.2 Universally Unique Identifiers method

For larger deployments, hierarchical allocation may be desired. The techniques in Universally Unique Identifier (UUID) URN ([RFC4122]) can be adapted for allocating unique object identifiers in sixty bits. An identifier is split into two components: a registrar prefix and sub-identifier. The registrar prefix defines an identifier block which is managed by an agent, the sub-identifier is a unique value within the registrar block.

For instance, each host in a network could be an agent so that unique identifiers for objects could be created autonomously be the host.

The identifier might be composed of a twenty-four bit host identifier followed by a thirty-six bit timestamp. Assuming that a host can allocate up to 100 identifiers per second, this allows about 21.8 years before wrap around.



Appendix C: Datacenter task virtualization

This section describes some details to apply ILA to virtualizing tasks in a datacenter.

C.1 Address per task

Managing the port number space for services within a datacenter is a nontrivial problem. When a service task is created, it may run on arbitrary hosts. The typical scenario is that the task will be started on some machine and will be assigned a port number for its service. The port number must be chosen dynamically to not conflict with any other port numbers already assigned to tasks on the same machine (possibly even other instances of the same service). A canonical name for the service is entered into a database with the host address and assigned port. When a client wishes to connect to the service, it queries the database with the service name to get both the address of an instance as well as its port number. Note that DNS is not adequate for the service lookup since it does not provide port numbers.

With ILA, each service task can be assigned its own IPv6 address and therefore will logically be assigned the full port space for that address. This a dramatic simplification since each service can now use a publicly known port number that does not need to unique between services or instances. A client can perform a lookup on the service name to get an IP address of an instance and then connect to that address using a well known port number. In this case, DNS is sufficient for directing clients to instances of a service.

C.2 Job scheduling

In the usual datacenter model, jobs are scheduled to run as tasks on some number of machines. A distributed job scheduler provides the scheduling which may entail considerable complexity since jobs will often have a variety of resource constraints. The scheduler takes these constraints into account while trying to maximize utility of

the datacenter in terms utilization, cost, latency, etc. Datacenter jobs do not typically run in virtual machines (VMs), but may run within containers. Containers are mechanisms that provide resource isolation between tasks running on the same host OS. These resources can include CPU, disk, memory, and networking.

A fundamental problem arises in that once a task for a job is scheduled on a machine, it often needs to run to completion. If the scheduler needs to schedule a higher priority job or change resource allocations, there may be little recourse but to kill tasks and restart them on a different machine. In killing a task, progress is lost which results in increased latency and wasted CPU cycles. Some tasks may checkpoint progress to minimize the amount of progress lost, but this is not a very transparent or general solution.

An alternative approach is to allow transparent job migration. The scheduler may migrate running jobs from one machine to another.

C.3 Task migration

Under the orchestration of the job scheduler, the steps to migrate a job may be:

1) Stop running tasks for the job.
2) Package the runtime state of the job. The runtime state is derived from the containers for the jobs.
3) Send the runtime state of the job to the new machine where the job is to run.
4) Instantiate the job’s state on the new machine.
5) Start the tasks for the job continuing from the point at which it was stopped.

This model similar to virtual machine (VM) migration except that the runtime state is typically much less data-- just task state as opposed to a full OS image. Task state may be compressed to reduce latency in migration.

C.3.1 Address migration

ILA facilitates address (specifically SIR address) migration between hosts as part of task migration or for other purposes. The steps in migrating an address might be:

1) Configure address on the target host.

2) Suspend use of the address on the old host. This includes handling established connections (see next section). A state may be established to drop packets or send ICMP destination

unreachable when packets to the migrated address are received.

3) Update the identifier to locator mapping database. Depending on the control plane implementation this may include pushing the new mapping to hosts.

4) Communicating hosts will learn of the new mapping via a control plane either by participation in a protocol for mapping propagation or by the ILA resolution protocol.

C.3.2 Connection migration

When a task and its addresses are migrated between machines, the disposition of existing TCP connections needs to be considered.

The simplest course of action is to drop TCP connections across a migration. Since migrations should be relatively rare events, it is conceivable that TCP connections could be automatically closed in the network stack during a migration event. If the applications running are known to handle this gracefully (i.e. reopen dropped connections) then this may be viable.

For seamless migration, open connections may be migrated between hosts. Migration of these entails pausing the connection, packaging connection state and sending to target, instantiating connection state in the peer stack, and restarting the connection. From the time the connection is paused to the time it is running again in the new stack, packets received for the connection should be silently dropped. For some period of time, the old stack will need to keep a record of the migrated connection. If it receives a packet, it should either silently drop the packet or forward it to the new location.

Author's Address

Tom Herbert
Quantonium
4701 Patrick Henry
Santa Clara, CA
EMail: tom@herbertland.com

Petr Lapukhov
1 Hacker Way
Menlo Parck, CA
EMail: petr@fb.com

V6OPS Working Group
Internet-Draft
Intended status: Informational
Expires: May 17, 2017

P. Matthews
Nokia
V. Kuarsingh
Cisco

November 13, 2016

Routing-Related Design Choices for IPv6 Networks
draft-ietf-v6ops-design-choices-12

Abstract

This document presents advice on certain routing-related design choices that arise when designing IPv6 networks (both dual-stack and IPv6-only). The intended audience is someone designing an IPv6 network who is knowledgeable about best current practices around IPv4 network design, and wishes to learn the corresponding practices for IPv6.

Table of Contents

1. Introduction 2
2. Design Choices 3
2.1. Addresses 3
2.1.1. Where to Use Addresses 4
2.1.2. Which Addresses to Use 6
2.2. Interfaces 7
2.2.1. Mix IPv4 and IPv6 on the Same Layer-3 Interface? . . 7
2.3. Static Routes 8
2.3.1. Link-Local Next-Hop in a Static Route? 8
2.4. IGPs 9
2.4.1. IGP Choice 9
2.4.2. IS-IS Topology Mode 12
2.4.3. RIP / RIPng 13
2.5. BGP 14
2.5.1. Which Transport for Which Routes? 14
2.5.1.1. BGP Sessions for Unlabeled Routes 16
2.5.1.2. BGP sessions for Labeled or VPN Routes 17
2.5.2. eBGP Endpoints: Global or Link-Local Addresses? . . . 18
3. General Observations 19
3.1. Use of Link-Local Addresses 19
3.2. Separation of IPv4 and IPv6 20
4. IANA Considerations 20
5. Security Considerations 20
6. Acknowledgements 21
7. Informative References 21
Authors' Addresses 25

1. Introduction

This document discusses routing-related design choices that arise when designing an IPv6-only or dual-stack network. The focus is on choices that do not come up when designing an IPv4-only network. The document presents each choice and the alternatives, and then discusses the pros and cons of the alternatives in detail. Where consensus currently exists around the best practice, this is documented; otherwise the document simply summarizes the current state of the discussion. Thus this document serves to both document the reasoning behind best current practices for IPv6, and to allow a designer to make an informed choice where no such consensus exists.

The design choices presented apply to both Service Provider and Enterprise network environments. Where choices have selection criteria which differ between the Service Provider and the Enterprise

environment, this is noted. The designer is encouraged to ensure that they familiarize themselves with any of the discussed technologies to ensure the best selection is made for their environment.

This document does not present advice on strategies for adding IPv6 to a network, nor does it discuss transition in these areas, see [RFC6180] for general advice,[RFC6782] for wireline service providers, [RFC6342]for mobile network providers, [RFC5963] for exchange point operators, [RFC6883] for content providers, and both [RFC4852] and [RFC7381] for enterprises. Nor does this document discuss the particulars of creating an IPv6 addressing plan; for advice in this area, see [RFC5375] or [v6-addressing-plan]. The document focuses on unicast routing design only and does not cover multicast or the issues involved in running MPLS over IPv6 transport

Section 2 presents and discusses a number of design choices. Section 3 discusses some general themes that run through these choices.

2. Design Choices

Each subsection below presents a design choice and discusses the pros and cons of the various options. If there is consensus in the industry for a particular option, then the consensus position is noted.

2.1. Addresses

This section discusses the choice of addresses for router loopbacks and links between routers. It does not cover the choice of addresses for end hosts.

In IPv6, an interface is always assigned a Link-Local Address (LLA) [RFC4291]. The link-local address can only be used for communicating with devices that are on-link, so often one or more additional addresses are assigned which are able to communicate off-link. This additional address or addresses can be one of three types:

- Provider-Independent Global Unicast Address (PI GUA): IPv6 address allocated by a regional address registry [RFC4291]
- Provider-Aggregatable Global Unicast Address (PA GUA): IPv6 Address allocated by your upstream service provider
- Unique Local Address (ULA): IPv6 address locally assigned [RFC4193]

This document uses the term "multi-hop address" to collectively refer to these three types of addresses.

PI GUAs are, for many situations, the most flexible of these choices. Their main disadvantages are that a regional address registry will only allocate them to organizations that meet certain qualifications, and one must pay an annual fee. These disadvantages mean that many smaller organization may not qualify or be willing to pay for these addresses.

PA GUAs have the advantage that they are usually provided at no extra charge when you contract with an upstream provider. However, they have the disadvantage that, when switching upstream providers, one must give back the old addresses and get new addresses from the new provider ("renumbering"). Though IPv6 has mechanisms to make renumbering easier than IPv4, these techniques are not generally applicable to routers and renumbering is still fairly hard [RFC5887] [RFC6879] [RFC7010] . PA GUAs also have the disadvantage that it is not easy to have multiple upstream providers ("multi-homing") if they are used (see "Ingress Filtering Problem" in [RFC5220]).

ULAs have the advantage that they are extremely easy to obtain and cost nothing. However, they have the disadvantage that they cannot be routed on the Internet, so must be used only within a limited scope. In many situations, this is not a problem, but in certain situations this can be problematic. Though there is currently no document that describes these situations, many of them are similar to those described in [RFC6752]. See also [I-D.ietf-v6ops-ula-usage-recommendations].

Not discussed in this document is the possibility of using the technology described in [RFC6296] to work around some of the limitations of PA GUAs and ULAs.

2.1.1. Where to Use Addresses

As mentioned above, all interfaces in IPv6 always have a link-local address. This section addresses the question of when and where to assign multi-hop addresses in addition to the LLA. We consider four options:

a. Use only link-local addresses on all router interfaces.

b. Assign multi-hop addresses to all link interfaces on each router, and use only a link-local address on the loopback interfaces.

c. Assign multi-hop addresses to the loopback interface on each router, and use only a link-local address on all link interfaces.

d. Assign multi-hop addresses to both link and loopback interfaces on each router.

Option (a) means that the router cannot be reached (ping, management, etc.) from farther than one-hop away. The authors are not aware of anyone using this option.

Option (b) means that the loopback interfaces are effectively useless, since link-local addresses cannot be used for the purposes that loopback interfaces are usually used for. So option (b) degenerates into option (d).

Thus the real choice comes down to option (c) vs. option (d).

Option (c) has two advantages over option (d). The first advantage is ease of configuration. In a network with a large number of links, the operator can just assign one multi-hop address to each router and then enable the IGP, without going through the tedious process of assigning and tracking the addresses on each link. The second advantage is security. Since packets with link-local addresses cannot be should not be routed, it is very difficult to attack the associated nodes from an off-link device. This implies less effort around maintaining security ACLs.

Countering these advantages are various disadvantages to option (c) compared with option (d):

- It is not possible to ping a link-local-only interface from a device that is not directly attached to the link. Thus, to troubleshoot, one must typically log into a device that is directly attached to the device in question, and execute the ping from there.

- A traceroute passing over the link-local-only interface will return the loopback address of the router, rather than the address of the interface itself.

- In cases of parallel point to point links it is difficult to determine which of the parallel links was taken when attempting to troubleshoot unless one sends packets directly between the two attached link-locals on the specific interfaces. Since many network problems behave differently for traffic to/from a router than for traffic through the router(s) in question, this can pose a significant hurdle to some troubleshooting scenarios.

- On some routers, by default the link-layer address of the interface is derived from the MAC address assigned to interface. When this is done, swapping out the interface hardware (e.g.

interface card) will cause the link-layer address to change. In some cases (peering config, ACLs, etc) this may require additional changes. However, many devices allow the link-layer address of an interface to be explicitly configured, which avoids this issue. This problem should fade away over time as more and more routers select interface identifiers according to the rules in [RFC7217].

- o The practice of naming router interfaces using DNS names is difficult and not recommended when using link-locals only. More generally, it is not recommended to put link-local addresses into DNS; see [RFC4472].

- o It is often not possible to identify the interface or link (in a database, email, etc) by giving just its address without also specifying the link in some manner.

It should be noted that it is quite possible for the same link-local address to be assigned to multiple interfaces. This can happen because the MAC address is duplicated (due to manufacturing process defaults or the use of virtualization), because a device deliberately re-uses automatically-assigned link-local addresses on different links, or because an operator manually assigns the same easy-to-type link-local address to multiple interfaces. All these are allowed in IPv6 as long as the addresses are used on different links.

For more discussion on the pros and cons, see [RFC7404]. See also [RFC5375] for IPv6 unicast address assignment considerations.

Today, most operators use option (d).

2.1.2. Which Addresses to Use

Having considered above whether or not to use a "multi-hop address", we now consider which of the addresses to use.

When selecting between these three "multi-hop address" types, one needs to consider exactly how they will be used. An important consideration is how Internet traffic is carried across the core of the network. There are two main options: (1) the classic approach where Internet traffic is carried as unlabeled traffic hop-by-hop across the network, and (2) the more recent approach where Internet traffic is carried inside an MPLS LSP (typically as part of a L3 VPN).

Under the classic approach:

- o PI GUAs are a very reasonable choice, if they are available.

- PA GUAs suffer from the "must renumber" and "difficult to multi-home" problems mentioned above.

- ULAs suffer from the "may be problematic" issues described above.

Under the MPLS approach:

- PA GUAs are a reasonable choice, if they are available.

- PA GUAs suffer from the "must renumber" problem, but the "difficult to multi-home" problem does not apply.

- ULAs are a reasonable choice, since (unlike in the classic approach) these addresses are not visible to the Internet, so the problematic cases do not occur.

2.2. Interfaces

2.2.1. Mix IPv4 and IPv6 on the Same Layer-3 Interface?

If a network is going to carry both IPv4 and IPv6 traffic, as many networks do today, then a question arises: Should an operator mix IPv4 and IPv6 traffic or keep them separated? More specifically, should the design:

a. Mix IPv4 and IPv6 traffic on the same layer-3 interface, OR

b. Separate IPv4 and IPv6 by using separate interfaces (e.g., two physical links or two VLANs on the same link)?

Option (a) implies a single layer-3 interface at each end of the connection with both IPv4 and IPv6 addresses; while option (b) implies two layer-3 interfaces at each end, one for IPv4 addresses and one with IPv6 addresses.

The advantages of option (a) include:

- Requires only half as many layer 3 interfaces as option (b), thus providing better scaling;

- May require fewer physical ports, thus saving money and simplifying operations;

- Can make the QoS implementation much easier (for example, rate-limiting the combined IPv4 and IPv6 traffic to or from a customer);

- Works well in practice, as any increase in IPv6 traffic is usually counter-balanced by a corresponding decrease in IPv4 traffic to or from the same host (ignoring the common pattern of an overall increase in Internet usage);

- And is generally conceptually simpler.

For these reasons, there is a relatively strong consensus in the operator community that option (a) is the preferred way to go. Most networks today use option (a) wherever possible.

However, there can be times when option (b) is the pragmatic choice. Most commonly, option (b) is used to work around limitations in network equipment. One big example is the generally poor level of support today for individual statistics on IPv4 traffic vs IPv6 traffic when option (a) is used. Other, device-specific, limitations exist as well. It is expected that these limitations will go away as support for IPv6 matures, making option (b) less and less attractive until the day that IPv4 is finally turned off.

2.3. Static Routes

2.3.1. Link-Local Next-Hop in a Static Route?

For the most part, the use of static routes in IPv6 parallels their use in IPv4. There is, however, one exception, which revolves around the choice of next-hop address in the static route. Specifically, should an operator:

a. Use the far-end's link-local address as the next-hop address, OR

b. Use the far-end's GUA/ULA address as the next-hop address?

Recall that the IPv6 specs for OSPF [RFC5340] and ISIS [RFC5308] dictate that they always use link-locals for next-hop addresses. For static routes, [RFC4861] section 8 says:

> A router MUST be able to determine the link-local address for each of its neighboring routers in order to ensure that the target address in a Redirect message identifies the neighbor router by its link-local address. For static routing, this requirement implies that the next-hop router's address should be specified using the link-local address of the router.

This implies that using a GUA or ULA as the next hop will prevent a router from sending Redirect messages for packets that "hit" this static route. All this argues for using a link-local as the next-hop address in a static route.

However, there are two cases where using a link-local address as the next-hop clearly does not work. One is when the static route is an indirect (or multi-hop) static route. The second is when the static route is redistributed into another routing protocol. In these cases, the above text from RFC 4861 notwithstanding, either a GUA or ULA must be used.

Furthermore, many network operators are concerned about the dependency of the default link-local address on an underlying MAC address, as described in the previous section.

Today most operators use GUAs as next-hop addresses.

2.4. IGPs

2.4.1. IGP Choice

One of the main decisions for a network operator looking to deploy IPv6 is the choice of IGP (Interior Gateway Protocol) within the network. The main options are OSPF, IS-IS and EIGRP. RIPng is another option, but very few networks run RIP in the core these days, so it is covered in a separate section below.

OSPF [RFC2328] [RFC5340] and IS-IS [RFC5120][RFC5120] are both standardized link-state protocols. Both protocols are widely supported by vendors, and both are widely deployed. By contrast, EIGRP [RFC7868] is a Cisco proprietary distance-vector protocol. EIGRP is rarely deployed in service-provider networks, but is quite common in enterprise networks, which is why it is discussed here.

It is out of scope for this document to describe all the differences between the three protocols; the interested reader can find books and websites that go into the differences in quite a bit of detail. Rather, this document simply highlights a few differences that can be important to consider when designing IPv6 or dual-stack networks.

Versions: There are two versions of OSPF: OSPFv2 and OSPFv3. The two versions share many concepts, are configured in a similar manner and seem very similar to most casual users, but have very different packet formats and other "under the hood" differences. The most important difference is that OSPFv2 will only route IPv4, while OSPFv3 will route both IPv4 and IPv6 (see [RFC5838]). OSPFv2 was by far the most widely deployed version of OSPF when this document was published. By contrast, both IS-IS and EIGRP have just a single version, which can route both IPv4 and IPv6.

Transport. IS-IS runs over layer 2 (e.g. Ethernet). This means that the functioning of IS-IS has no dependencies on the IP layer: if

there is a problem at the IP layer (e.g. bad addresses), two routers can still exchange IS-IS packets. By contrast, OSPF and EIGRP both run over the IP layer. This means that the IP layer must be configured and working OSPF or EIGRP packets to be exchanged between routers. For EIGRP, the dependency on the IP layer is simple: EIGRP for IPv4 runs over IPv4, while EIGRP for IPv6 runs over IPv6. For OSPF, the story is more complex: OSPFv2 runs over IPv4, but OSPFv3 can run over either IPv4 or IPv6. Thus it is possible to route both IPv4 and IPv6 with OSPFv3 running over IPv6 or with OSPFv3 running over IPv4. This means that there are number of choices for how to run OSPF in a dual-stack network:

- Use OSPFv2 for routing IPv4 , and OSPFv3 running over IPv6 for routing IPv6, OR
- Use OSPFv3 running over IPv6 for routing both IPv4 and IPv6, OR
- Use OSPFv3 running over IPv4 for routing both IPv4 and IPv6.

Summarization and MPLS: For most casual users, the three protocols are fairly similar in what they can do, with two glaring exceptions: summarization and MPLS. For summarization, both OSPF and IS-IS have the concept of summarization between areas, but the two area concepts are quite different, and an area design that works for one protocol will usually not work for the other. EIGRP has no area concept, but has the ability to summarize at any router. Thus a large network will typically have a very different OSPF, IS-IS and EIGRP designs, which is important to keep in mind if you are planning on using one protocol to route IPv4 and a different protocol for IPv6. The other difference is that OSPF and IS-IS both support RSVP-TE, a widely-used MPLS signaling protocol, while EIGRP does not: this is due to OSPF and IS-IS both being link-state protocols while EIGRP is a distance-vector protocol.

The table below sets out possible combinations of protocols to route both IPv4 and IPv6, and makes some observations on each combination. Here "EIGRP-v4" means "EIGRP for IPv4" and similarly for "EIGRP-v6". For OSPFv3, it is possible to run it over either IPv4 or IPv6; this is not indicated in the table.

IGP for IPv4	IGP for IPv6	Protocol separation	Similar configuration possible	Multiple Known Deployments
OSPFv2	OSPFv3	YES	YES	YES (8)
OSPFv2	IS-IS	YES	-	YES (3)
OSPFv2	EIGRP-v6	YES	-	-
OSPFv3	OSPFv3	NO	YES	-
OSPFv3	IS-IS	YES	-	-
OSPFv3	EIGRP-v6	YES	-	-
IS-IS	OSPFv3	YES	-	YES (2)
IS-IS	IS-IS	-	YES	YES (12)
IS-IS	EIGRP-v6	YES	-	-
EIGRP-v4	OSPFv3	YES	-	? (1)
EIGRP-v4	IS-IS	YES	-	-
EIGRP-v4	EIGRP-v6	-	YES	? (2)

In the column "Multiple Known Deployments", a YES indicates that a significant number of production networks run this combination, with the number of such networks indicated in parentheses following, while a "?" indicates that the authors are only aware of one or two small networks that run this combination. Data for this column was gathered from an informal poll of operators on a number of mailing lists. This poll was not intended to be a thorough scientific study of IGP choices, but to provide a snapshot of known operator choices at the time of writing (Mid-2015) for successful production dual stack network deployments. There were twenty six (26) network implementations represented by 17 respondents. Some respondents provided information on more then one network or network deployment. Due to privacy considerations, the networks' represented and respondents are not listed in this document.

A number of combinations are marked as offering "Protocol separation". These options use a different IGP protocol for IPv4 vs IPv6. With these options, a problem with routing IPv6 is unlikely to affect IPv4 or visa-versa. Some operator may consider this as a benefit when first introducing dual stack capabilities or for ongoing technical reasons.

Three combinations are marked "Similar configuration possible". This means it is possible (but not required) to use very similar IGP configuration for IPv4 and IPv6: for example, the same area boundaries, area numbering, link costing, etc. If you are happy with your IPv4 IGP design, then this will likely be a consideration. By contrast, the options that use, for example, IS-IS for one IP version and OSPF for the other version will require considerably different configuration, and will also require the operations staff to become familiar with the difference between the two protocols.

It should be noted that a number of ISPs have run OSPF as their IPv4 IGP for quite a few years, but have selected IS-IS as their IPv6 IGP. However, there are very few (none?) that have made the reverse choice. This is, in part, because routers generally support more nodes in an IS-IS area than in the corresponding OSPF area, and because IS-IS is seen as more secure because it runs at layer 2.

2.4.2. IS-IS Topology Mode

When IS-IS is used to route both IPv4 and IPv6, then there is an additional choice of whether to run IS-IS in single-topology or multi-topology mode.

With single-topology mode (also known as Native mode) [RFC5308]:

- IS-IS keeps a single link-state database for both IPv4 and IPv6.
- There is a single set of link costs which apply to both IPv4 and IPv6.
- All links in the network must support both IPv4 and IPv6, as the calculation of routes does not take this into account. If some links do not support IPv6 (or IPv4), then packets may get routed across links where support is lacking and get dropped. This can cause problems if some network devices do not support IPv6 (or IPv4).
- It is also important to keep the previous point in mind when adding or removing support for either IPv4 or IPv6.

With multi-topology mode [RFC5120]:

- IS-IS keeps two link-state databases, one for IPv4 and one for IPv6.

- IPv4 and IPv6 can have separate link metrics. Note that most implementations today require separate link metrics: a number of operators have rudely discovered that they have forgotten to configure the IPv6 metric until sometime after deploying IPv6 in multi-topology mode!

- Some links can be IPv4-only, some IPv6-only, and some dual-stack. Routes to IPv4 and IPv6 addresses are computed separately and may take different paths even if the addresses are located on the same remote device.

- The previous point may help when adding or removing support for either IPv4 or IPv6.

In the informal poll of operators, out of 12 production networks that ran IS-IS for both IPv4 and IPv6, 6 used single topology mode, 4 used multi-topology mode, and 2 did not specify. One motivation often cited by then operators for using Single Topology mode was because some device did not support multi-topology mode.

When asked, many people feel multi-topology mode is superior to single-topology mode because it provides greater flexibility at minimal extra cost. Never-the-less, as shown by the poll results, a number of operators have used single-topology mode successfully.

Note that this issue does not come up with OSPF, since there is nothing that corresponds to IS-IS single-topology mode with OSPF.

2.4.3. RIP / RIPng

A protocol option not described in the table above is RIP for IPv4 and RIPng for IPv6 [RFC2080]. These are distance vector protocols that are almost universally considered to be inferior to OSPF, IS-IS, or EIGRP for general use.

However, there is one specialized use where RIP/RIPng is still considered to be appropriate: in star topology networks where a single core device has lots and lots of links to edge devices and each edge device has only a single path back to the core. In such networks, the single path means that the limitations of RIP/RIPng are mostly not relevant and the very light-weight nature of RIP/RIPng gives it an advantage over the other protocols mentioned above. One concrete example of this scenario is the use of RIP/RIPng between cable modems and the CMTS.

2.5. BGP

2.5.1. Which Transport for Which Routes?

BGP these days is multi-protocol. It can carry routes of many different types, or more precisely, many different AFI/SAFI combinations. It can also carry routes when the BGP session, or more accurately the underlying TCP connection, runs over either IPv4 or IPv6 (here referred to as either "IPv4 transport" or "IPv6 transport"). Given this flexibility, one of the biggest questions when deploying BGP in a dual-stack network is the question of which route types should be carried over sessions using IPv4 transport and which should be carried over sessions using IPv6 transport.

This section discusses this question for the three most-commonly-used SAFI values: unlabeled (SAFI 1), labeled (SAFI 4) and VPN (SAFI 128). Though we do not explicitly discuss other SAFI values, many of the comments here can be applied to the other values.

Consider the following table:

Route Family	Transport	Comments
Unlabeled IPv4	IPv4	Works well
Unlabeled IPv4	IPv6	Next-hop
Unlabeled IPv6	IPv4	Next-hop
Unlabeled IPv6	IPv6	Works well
Labeled IPv4	IPv4	Works well
Labeled IPv4	IPv6	Next-hop
Labeled IPv6	IPv4	(6PE) Works well
Labeled IPv6	IPv6	Next-hop or MPLS over IPv6
VPN IPv4	IPv4	Works well
VPN IPv4	IPv6	Next-hop
VPN IPv6	IPv4	(6VPE) Works well
VPN IPv6	IPv6	Next-hop or MPLS over IPv6

The first column in this table lists various route families, where "unlabeled" means SAFI 1, "labeled" means the routes carry an MPLS label (SAFI 4, see [RFC3107]), and "VPN" means the routes are normally associated with a layer-3 VPN (SAFI 128, see [RFC4364]). The second column lists the protocol used to transport the BGP session, frequently specified by giving either an IPv4 or IPv6 address in the "neighbor" statement.

The third column comments on the combination in the first two columns:

- For combinations marked "Works well", these combinations are standardized, widely supported and widely deployed.

- For combinations marked "Next-hop", these combinations are not standardized and are less-widely supported. These combinations all have the "next-hop mismatch" problem: the transported route needs a next-hop address from the other address family than the transport address (for example, an IPv4 route needs an IPv4 next-hop, even when transported over IPv6). Some vendors have implemented ways to solve this problem for specific combinations, but for combinations marked "next-hop", these solutions have not been standardized (cf. 6PE and 6VPE, where the solution has been standardized).

- For combinations marked as "Next-hop or MPLS over IPv6", these combinations either require a non-standard solution to the next-hop problem, or require MPLS over IPv6. At the time of writing, MPLS over IPv6 is not widely supported or deployed.

Also, it is important to note that changing the set of address families being carried over a BGP session requires the BGP session to be reset (unless something like [I-D.ietf-idr-dynamic-cap] or [I-D.ietf-idr-bgp-multisession] is in use). This is generally more of an issue with eBGP sessions than iBGP sessions: for iBGP sessions it is common practice for a router to have two iBGP sessions, one to each member of a route reflector pair, so one can change the set of address families on first one of the sessions and then the other.

The following subsections discuss specific combinations in more detail.

2.5.1.1. BGP Sessions for Unlabeled Routes

Unlabeled routes are commonly carried on eBGP sessions, as well as on iBGP sessions in networks where Internet traffic is carried unlabeled across the network.

In these scenarios, there are three reasonable choices:

a. Carry unlabeled IPv4 and IPv6 routes over IPv4, OR

b. Carry unlabeled IPv4 and IPv6 routes over IPv6, OR

c. Carry unlabeled IPv4 routes over IPv4, and unlabeled IPv6 routes over IPv6

Options (a) and (b) have the advantage that one one BGP session is required between pairs of routers. However, option (c) is widely considered to be the best choice. There are several reasons for this :

- o It gives a clean separation between IPv4 and IPv6. This can be especially useful when first deploying IPv6 and troubleshooting resulting problems.

- o This avoids the next-hop problem described above.

- o The status of the routes follows the status of the underlying transport. If, for example, the IPv6 data path between the two BGP speakers fails, then the IPv6 session between the two speakers will fail and the IPv6 routes will be withdrawn, which will allow the traffic to be re-routed elsewhere. By contrast, if the IPv6 routes were transported over IPv4, then the failure of the IPv6 data path might leave a working IPv4 data path, so the BGP session would remain up and the IPv6 routes would not be withdrawn, and thus the IPv6 traffic would be sent into a black hole.

- o It avoids resetting the BGP session when adding IPv6 to an existing session, or when removing IPv4 from an existing session.

Rarely, there are situations where option (c) is not practical. In those cases today, most operators use option (a), carrying both route types over a single BGP session.

2.5.1.2. BGP sessions for Labeled or VPN Routes

When carrying labeled or VPN routes, the only widely-supported solution at time of writing is to carry both route types over IPv4. This may change in as MPLS over IPv6 becomes more widely implemented.

There are two options when carrying both over IPv4:

a. Carry all routes over a single BGP session, OR

b. Carry the routes over multiple BGP sessions (e.g. one for VPN IPv4 routes and one for VPN IPv6 routes)

Using a single session is usually simplest for an iBGP session going to a route reflector handling both route families. Using a single session here usually means that the BGP session will reset when changing the set of address families, but as noted above, this is usually not a problem when redundant route reflectors are involved.

In eBGP situations, two sessions are usually more appropriate. [JUSTIFICATION?]

2.5.2. eBGP Endpoints: Global or Link-Local Addresses?

When running eBGP over IPv6, there are two options for the addresses to use at each end of the eBGP session (or more properly, the underlying TCP session):

a. Use link-local addresses for the eBGP session, OR

b. Use global addresses for the eBGP session.

Note that the choice here is the addresses to use for the eBGP sessions, and not whether the link itself has global (or unique-local) addresses. In particular, it is quite possible for the eBGP session to use link-local addresses even when the link has global addresses.

The big attraction for option (a) is security: an eBGP session using link-local addresses is extremely difficult to attack from a device that is off-link. This provides very strong protection against TCP RST and similar attacks. Though there are other ways to get an equivalent level of security (e.g. GTSM [RFC5082], MD5 [RFC5925], or ACLs), these other ways require additional configuration which can be forgotten or potentially mis-configured.

However, there are a number of small disadvantages to using link-local addresses:

- Using link-local addresses only works for single-hop eBGP sessions; it does not work for multi-hop sessions.
- One must use "next-hop self" at both endpoints, otherwise re-advertising routes learned via eBGP into iBGP will not work. (Some products enable "next-hop self" in this situation automatically).
- Operators and their tools are used to referring to eBGP sessions by address only, something that is not possible with link-local addresses.
- If one is configuring parallel eBGP sessions for IPv4 and IPv6 routes, then using link-local addresses for the IPv6 session introduces extra operational differences between the two sessions which could otherwise be avoided.
- On some products, an eBGP session using a link-local address is more complex to configure than a session that uses a global address.

- If hardware or other issues cause one to move the cable to a different local interface, then reconfiguration is required at both ends: at the local end because the interface has changed (and with link-local addresses, the interface must always be specified along with the address), and at the remote end because the link-local address has likely changed. (Contrast this with using global addresses, where less re-configuration is required at the local end, and no reconfiguration is required at the remote end).

- Finally, a strict application of [RFC2545] forbids running eBGP between link-local addresses, as [RFC2545] requires the BGP next-hop field to contain at least a global address.

For these reasons, most operators today choose to have their eBGP sessions use global addresses.

3. General Observations

There are two themes that run though many of the design choices in this document. This section presents some general discussion on these two themes.

3.1. Use of Link-Local Addresses

The proper use of link-local addresses is a common theme in the IPv6 network design choices. Link-layer addresses are, of course, always present in an IPv6 network, but current network design practice mostly ignores them, despite efforts such as [RFC7404].

There are three main reasons for this current practice:

- Network operators are concerned about the volatility of link-local addresses based on MAC addresses, despite the fact that this concern can be overcome by manually-configuring link-local addresses;

- It is very difficult to impossible to ping a link-local address from a device that is not on the same subnet. This is a troubleshooting disadvantage, though it can also be viewed as a security advantage.

- Most operators are currently running networks that carry both IPv4 and IPv6 traffic, and wish to harmonize their IPv4 and IPv6 design and operational practices where possible.

3.2. Separation of IPv4 and IPv6

Currently, most operators are running or planning to run networks that carry both IPv4 and IPv6 traffic. Hence the question: To what degree should IPv4 and IPv6 be kept separate? As can be seen above, this breaks into two sub-questions: To what degree should IPv4 and IPv6 traffic be kept separate, and to what degree should IPv4 and IPv6 routing information be kept separate?

The general consensus around the first question is that IPv4 and IPv6 traffic should generally be mixed together. This recommendation is driven by the operational simplicity of mixing the traffic, plus the general observation that the service being offered to the end user is Internet connectivity and most users do not know or care about the differences between IPv4 and IPv6. Thus it is very desirable to mix IPv4 and IPv6 on the same link to the end user. On other links, separation is possible but more operationally complex, though it does occasionally allow the operator to work around limitations on network devices. The situation here is roughly comparable to IP and MPLS traffic: many networks mix the two traffic types on the same links without issues.

By contrast, there is more of an argument for carrying IPv6 routing information over IPv6 transport, while leaving IPv4 routing information on IPv4 transport. By doing this, one gets fate-sharing between the control and data plane for each IP protocol version: if the data plane fails for some reason, then often the control plane will too.

4. IANA Considerations

This document makes no requests of IANA.

5. Security Considerations

This document introduces no new security considerations that are not already documented elsewhere.

The following is a brief list of pointers to documents related to the topics covered above that the reader may wish to review for security considerations.

For general IPv6 security, [RFC4942] provides guidance on security considerations around IPv6 transition and coexistence.

For OSPFv3, the base protocol specification [RFC5340] has a short security considerations section which notes that the fundamental

mechanism for protecting OSPFv3 from attacks is the mechanism described in [RFC4552].

For IS-IS, [RFC5308] notes that ISIS for IPv6 raises no new security considerations over ISIS for IPv4 over those documented in [ISO10589] and [RFC5304].

For BGP, [RFC2545] notes that BGP for IPv6 raises no new security considerations over those present in BGP for IPv4. However, there has been much discussion of BGP security recently, and the interested reader is referred to the documents of the IETF's SIDR working group.

6. Acknowledgements

Many, many people in the V6OPS working group provided comments and suggestions that made their way into this document. A partial list includes: Rajiv Asati, Fred Baker, Michael Behringer, Marc Blanchet, Ron Bonica, Randy Bush, Cameron Byrne, Brian Carpenter, KK Chittimaneni, Tim Chown, Lorenzo Colitti, Gert Doering, Francis Dupont, Bill Fenner, Kedar K Gaonkar, Chris Grundemann, Steinar Haug, Ray Hunter, Joel Jaeggli, Victor Kuarsingh, Jen Linkova, Ivan Pepelnjak, Alexandru Petrescu, Rob Shakir, Mark Smith, Jean-Francois Tremblay, Dave Thaler, Tina Tsou, Eric Vyncke, Dan York, and Xuxiaohu.

The authors would also like to thank Pradeep Jain and Alastair Johnson for helpful comments on a very preliminary version of this document.

7. Informative References

[I-D.ietf-idr-bgp-multisession]
Scudder, J., Appanna, C., and I. Varlashkin, "Multisession BGP", draft-ietf-idr-bgp-multisession-07 (work in progress), September 2012.

[I-D.ietf-idr-dynamic-cap]
Ramachandra, S. and E. Chen, "Dynamic Capability for BGP-4", draft-ietf-idr-dynamic-cap-14 (work in progress), December 2011.

[I-D.ietf-v6ops-ula-usage-recommendations]
Liu, B. and S. Jiang, "Considerations For Using Unique Local Addresses", draft-ietf-v6ops-ula-usage-recommendations-05 (work in progress), May 2015.

[ISO10589]
International Standards Organization, "Intermediate system to Intermediate system intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473)", International Standard 10589:2002, Nov 2002.

[RFC2080] Malkin, G. and R. Minnear, "RIPng for IPv6", RFC 2080, DOI 10.17487/RFC2080, January 1997, <http://www.rfc-editor.org/info/rfc2080>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <http://www.rfc-editor.org/info/rfc2328>.

[RFC2545] Marques, P. and F. Dupont, "Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing", RFC 2545, DOI 10.17487/RFC2545, March 1999, <http://www.rfc-editor.org/info/rfc2545>.

[RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, DOI 10.17487/RFC3107, May 2001, <http://www.rfc-editor.org/info/rfc3107>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <http://www.rfc-editor.org/info/rfc4193>.

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

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <http://www.rfc-editor.org/info/rfc4364>.

[RFC4472] Durand, A., Ihren, J., and P. Savola, "Operational Considerations and Issues with IPv6 DNS", RFC 4472, DOI 10.17487/RFC4472, April 2006, <http://www.rfc-editor.org/info/rfc4472>.

[RFC4552] Gupta, M. and N. Melam, "Authentication/Confidentiality for OSPFv3", RFC 4552, DOI 10.17487/RFC4552, June 2006, <http://www.rfc-editor.org/info/rfc4552>.

[RFC4852] Bound, J., Pouffary, Y., Klynsma, S., Chown, T., and D. Green, "IPv6 Enterprise Network Analysis - IP Layer 3 Focus", RFC 4852, DOI 10.17487/RFC4852, April 2007, <http://www.rfc-editor.org/info/rfc4852>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC4942] Davies, E., Krishnan, S., and P. Savola, "IPv6 Transition/ Co-existence Security Considerations", RFC 4942, DOI 10.17487/RFC4942, September 2007, <http://www.rfc-editor.org/info/rfc4942>.

[RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., Ed., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, DOI 10.17487/RFC5082, October 2007, <http://www.rfc-editor.org/info/rfc5082>.

[RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, DOI 10.17487/RFC5120, February 2008, <http://www.rfc-editor.org/info/rfc5120>.

[RFC5220] Matsumoto, A., Fujisaki, T., Hiromi, R., and K. Kanayama, "Problem Statement for Default Address Selection in Multi-Prefix Environments: Operational Issues of RFC 3484 Default Rules", RFC 5220, DOI 10.17487/RFC5220, July 2008, <http://www.rfc-editor.org/info/rfc5220>.

[RFC5304] Li, T. and R. Atkinson, "IS-IS Cryptographic Authentication", RFC 5304, DOI 10.17487/RFC5304, October 2008, <http://www.rfc-editor.org/info/rfc5304>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <http://www.rfc-editor.org/info/rfc5308>.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <http://www.rfc-editor.org/info/rfc5340>.

[RFC5375] Van de Velde, G., Popoviciu, C., Chown, T., Bonness, O., and C. Hahn, "IPv6 Unicast Address Assignment Considerations", RFC 5375, DOI 10.17487/RFC5375, December 2008, <http://www.rfc-editor.org/info/rfc5375>.

[RFC5838] Lindem, A., Ed., Mirtorabi, S., Roy, A., Barnes, M., and R. Aggarwal, "Support of Address Families in OSPFv3", RFC 5838, DOI 10.17487/RFC5838, April 2010, <http://www.rfc-editor.org/info/rfc5838>.

[RFC5887] Carpenter, B., Atkinson, R., and H. Flinck, "Renumbering Still Needs Work", RFC 5887, DOI 10.17487/RFC5887, May 2010, <http://www.rfc-editor.org/info/rfc5887>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <http://www.rfc-editor.org/info/rfc5925>.

[RFC5963] Gagliano, R., "IPv6 Deployment in Internet Exchange Points (IXPs)", RFC 5963, DOI 10.17487/RFC5963, August 2010, <http://www.rfc-editor.org/info/rfc5963>.

[RFC6180] Arkko, J. and F. Baker, "Guidelines for Using IPv6 Transition Mechanisms during IPv6 Deployment", RFC 6180, DOI 10.17487/RFC6180, May 2011, <http://www.rfc-editor.org/info/rfc6180>.

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <http://www.rfc-editor.org/info/rfc6296>.

[RFC6342] Koodli, R., "Mobile Networks Considerations for IPv6 Deployment", RFC 6342, DOI 10.17487/RFC6342, August 2011, <http://www.rfc-editor.org/info/rfc6342>.

[RFC6752] Kirkham, A., "Issues with Private IP Addressing in the Internet", RFC 6752, DOI 10.17487/RFC6752, September 2012, <http://www.rfc-editor.org/info/rfc6752>.

[RFC6782] Kuarsingh, V., Ed. and L. Howard, "Wireline Incremental IPv6", RFC 6782, DOI 10.17487/RFC6782, November 2012, <http://www.rfc-editor.org/info/rfc6782>.

[RFC6879] Jiang, S., Liu, B., and B. Carpenter, "IPv6 Enterprise Network Renumbering Scenarios, Considerations, and Methods", RFC 6879, DOI 10.17487/RFC6879, February 2013, <http://www.rfc-editor.org/info/rfc6879>.

[RFC6883] Carpenter, B. and S. Jiang, "IPv6 Guidance for Internet Content Providers and Application Service Providers", RFC 6883, DOI 10.17487/RFC6883, March 2013, <http://www.rfc-editor.org/info/rfc6883>.

[RFC7010] Liu, B., Jiang, S., Carpenter, B., Venaas, S., and W. George, "IPv6 Site Renumbering Gap Analysis", RFC 7010, DOI 10.17487/RFC7010, September 2013, <http://www.rfc-editor.org/info/rfc7010>.

[RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <http://www.rfc-editor.org/info/rfc7217>.

[RFC7381] Chittimaneni, K., Chown, T., Howard, L., Kuarsingh, V., Pouffary, Y., and E. Vyncke, "Enterprise IPv6 Deployment Guidelines", RFC 7381, DOI 10.17487/RFC7381, October 2014, <http://www.rfc-editor.org/info/rfc7381>.

[RFC7404] Behringer, M. and E. Vyncke, "Using Only Link-Local Addressing inside an IPv6 Network", RFC 7404, DOI 10.17487/RFC7404, November 2014, <http://www.rfc-editor.org/info/rfc7404>.

[RFC7868] Savage, D., Ng, J., Moore, S., Slice, D., Paluch, P., and R. White, "Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)", RFC 7868, DOI 10.17487/RFC7868, May 2016, <http://www.rfc-editor.org/info/rfc7868>.

[v6-addressing-plan] SurfNet, "Preparing an IPv6 Address Plan", 2013, <http://www.ripe.net/lir-services/training/material/IPv6-for-LIRs-Training-Course/Preparing-an-IPv6-Addressing-Plan.pdf>.

Authors' Addresses

Philip Matthews
Nokia
600 March Road
Ottawa, Ontario K2K 2E6
Canada

Phone: +1 613-784-3139
Email: philip_matthews@magma.ca

Victor Kuarsingh
Cisco
88 Queens Quay
Toronto, ON M5J0B8
Canada

Email: victor@jvknet.com

IPv6 Operations
Internet-Draft
Intended status: Best Current Practice
Expires: November 26, 2016

L. Colitti
V. Cerf
Google
S. Cheshire
D. Schinazi
Apple Inc.

May 25, 2016

Host address availability recommendations
draft-ietf-v6ops-host-addr-availability-07

Abstract

This document recommends that networks provide general-purpose end hosts with multiple global IPv6 addresses when they attach, and describes the benefits of and the options for doing so.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 26, 2016.

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Requirements Language 3
2. Common IPv6 deployment model 3
3. Benefits of providing multiple addresses 3
4. Problems with restricting the number of addresses per host . 4
5. Overcoming limits using Network Address Translation 5
6. Options for providing more than one address 6
7. Number of addresses required 7
8. Recommendations 8
9. Operational considerations 8
9.1. Host tracking 8
9.2. Address space management 9
9.3. Addressing link layer scalability issues via IP routing . 10
10. Acknowledgements 11
11. IANA Considerations 11
12. Security Considerations 11
13. References 11
13.1. Normative References 11
13.2. Informative References 11
Authors' Addresses 14

1. Introduction

In most aspects, the IPv6 protocol is very similar to IPv4. This similarity can create a tendency to think of IPv6 as 128-bit IPv4, and thus lead network designers and operators to apply identical configurations and operational practices to both. This is generally a good thing because it eases the transition to IPv6 and the operation of dual-stack networks. However, in some design and operational areas it can lead to carrying over IPv4 practices that are limiting or not appropriate in IPv6 due to differences between the protocols.

One such area is IP addressing, particularly IP addressing of hosts. This is substantially different because unlike IPv4 addresses, IPv6 addresses are not a scarce resource. In IPv6, a single link provides over four billion times more address space than the whole IPv4 Internet [RFC7421]. Thus, unlike IPv4, IPv6 networks are not forced by address availability considerations to provide only one address per host. On the other hand, providing multiple addresses has many benefits including application functionality and simplicity, privacy, flexibility to accommodate future applications, and the ability to

provide Internet access without the use of NAT. Providing only one IPv6 address per host negates these benefits.

This document describes the benefits of providing multiple addresses per host and the problems with not doing so. It recommends that networks provide general-purpose end hosts with multiple global addresses when they attach, and lists current options for doing so. It does not specify any changes to protocols or host behavior.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

2. Common IPv6 deployment model

IPv6 is designed to support multiple addresses, including multiple global addresses, per interface ([RFC4291] section 2.1, [RFC6434] section 5.9.4). Today, many general-purpose IPv6 hosts are configured with three or more addresses per interface: a link-local address, a stable address (e.g., using EUI-64 or Opaque Interface Identifiers [RFC7217]), one or more privacy addresses [RFC4941], and possibly one or more temporary or non-temporary addresses obtained using DHCPv6 [RFC3315].

In most general-purpose IPv6 networks, including all 3GPP networks ([RFC6459] section 5.2) and Ethernet and Wi-Fi networks using SLAAC [RFC4862], IPv6 hosts have the ability to configure additional IPv6 addresses from the link prefix(es) without explicit requests to the network.

3. Benefits of providing multiple addresses

Today, there are many host functions that require more than one IP address to be available to the host, including:

- Privacy addressing to prevent tracking by off-network hosts [RFC4941].
- Multiple processors inside the same device. For example, in many mobile devices both the application processor and baseband processor need to communicate with the network, particularly for technologies like I-WLAN [TS.24327] where the two processors share the Wi-Fi network connection.
- Extending the network (e.g., "tethering").

- Running virtual machines on hosts.

- Translation-based transition technologies such as 464XLAT [RFC6877] that provide IPv4 over IPv6. Some of these technologies require the availability of a dedicated IPv6 address in order to determine whether inbound packets are translated or native ([RFC6877] section 6.3).

- ILA ("Identifier-locator addressing") [I-D.herbert-nvo3-ila].

- Future applications (e.g., per-application IPv6 addresses [TARP]).

Examples of how the availability of multiple addresses per host has already allowed substantial deployment of new applications without explicit requests to the network are:

- 464XLAT. 464XLAT is usually deployed within a particular network, and in this model the operator can ensure that the network is appropriately configured to provide the CLAT with the additional IPv6 address it needs to implement 464XLAT. However, there are deployments where the PLAT (i.e., NAT64) is provided as a service by a different network, without the knowledge or cooperation of the residential ISP (e.g., the IPv6v4 Exchange Service <http://www.jpix.ad.jp/en/service/ipv6v4.html>). This type of deployment is only possible because those residential ISPs provide multiple IP addresses to their users, and thus those users can freely obtain the extra IPv6 address required to run 464XLAT.

- /64 sharing [RFC7278]. When the topology supports it, this is a way to provide IPv6 tethering without needing to wait for network operators to deploy DHCPv6 PD, which is only available in 3GPP release 10 or above ([RFC6459] section 5.3).

4. Problems with restricting the number of addresses per host

Providing a restricted number of addresses per host implies that functions that require multiple addresses will either be unavailable (e.g., if the network provides only one IPv6 address per host, or if the host has reached the limit of the number of addresses available), or that the functions will only be available after an explicit request to the network is granted. The necessity of explicit requests has the following drawbacks:

- Increased latency, because a provisioning operation, and possibly human intervention with an update to the service level agreement, must complete before the functionality is available.

- Uncertainty, because it is not known if a particular operation function will be available until the provisioning operation succeeds or fails.

- Complexity, because implementations need to deal with failures and somehow present them to the user. Failures may manifest as timeouts, which may be slow and frustrating to users.

- Increased load on the network's provisioning servers.

Some operators may desire to configure their networks to limit the number of IPv6 addresses per host. Reasons might include hardware limitations (e.g., TCAM or neighbor cache table size constraints), business models (e.g., a desire to charge the network's users on a per-device basis), or operational consistency with IPv4 (e.g., an IP address management system that only supports one address per host). However, hardware limitations are expected to ease over time, and an attempt to generate additional revenue by charging per device may prove counterproductive if customers respond (as they did with IPv4) by using NAT, which results in no additional revenue, but leads to more operational problems and higher support costs.

5. Overcoming limits using Network Address Translation

These limits can mostly be overcome by end hosts by using NAT, and indeed in IPv4 most of these functions are provided by using NAT on the host. Thus, the limits could be overcome in IPv6 as well by implementing NAT66 on the host.

Unfortunately NAT has well-known drawbacks. For example, it causes application complexity due to the need to implement NAT traversal. It hinders development of new applications. On mobile devices, it reduces battery life due to the necessity of frequent keepalives, particularly for UDP. Applications using UDP that need to work on most of the Internet are forced to send keepalives at least every 30 seconds <http://www.ietf.org/proceedings/88/slides/slides-88-tsvarea-10.pdf>. For example, the QUIC protocol uses a 15-second keepalive [I-D.tsvwg-quic-protocol]. Other drawbacks of NAT are well known and documented [RFC2993]. While IPv4 NAT is inevitable due to the limited amount of IPv4 space available, that argument does not apply to IPv6. Guidance from the IAB is that deployment of IPv6 NAT is not desirable [RFC5902].

The desire to overcome the problems listed in Section 4 without disabling any features has resulted in developers implementing IPv6 NAT. There are fully-stateful address+port NAT66 implementations in client operating systems today: for example, Linux has supported NAT66 since late 2012 <http://kernelnewbies.org/Linux_3.7#head-

103e14959eeb974bbd4e862df8afe7c118ba2beb>. A popular software hypervisor also recently implemented NAT66 to work around these issues <https://communities.vmware.com/docs/DOC-29954>. Wide deployment of networks that provide a restricted number of addresses will cause proliferation of NAT66 implementations.

This is not a desirable outcome. It is not desirable for users because they may experience application brittleness. It is likely not desirable for network operators either, as they may suffer higher support costs, and even when the decision to provide only one IPv6 address per device is dictated by the network's business model, there may be little in the way of incremental revenue, because devices can share their IPv6 address with other devices. Finally, it is not desirable for operating system manufacturers and application developers, who will have to build more complexity, lengthening development time and/or reducing the time spent on other features.

Indeed, it could be argued that the main reason for deploying IPv6, instead of continuing to scale the Internet using only IPv4 and large-scale NAT44, is because doing so can provide all the hosts on the planet with end-to-end connectivity that is constrained not by accidental technical limitations, but only by intentional security policies.

6. Options for providing more than one address

Multiple IPv6 addresses can be provided in the following ways:

- Using Stateless Address Autoconfiguration [RFC4862]. SLAAC allows hosts to create global IPv6 addresses on demand by simply forming new addresses from the global prefix(es) assigned to the link. Typically, SLAAC is used on shared links, but it is also possible to use SLAAC while providing a dedicated /64 prefix to each host. This is the case, for example, if the host is connected via a point-to-point link such as in 3GPP networks, on a network where each host has its own dedicated VLAN, or on a wireless network where every MAC address is placed in its own broadcast domain.

- Using stateful DHCPv6 address assignment [RFC3315]. Most DHCPv6 clients only ask for one non-temporary address, but the protocol allows requesting multiple temporary and even multiple non-temporary addresses, and the server could choose to provide multiple addresses. It is also technically possible for a client to request additional addresses using a different DUID, though the DHCPv6 specification implies that this is not expected behavior ([RFC3315] section 9). The DHCPv6 server will decide whether to grant or reject the request based on information about the client, including its DUID, MAC address, and so on. The maximum number of

IPv6 addresses that can be provided in a single DHCPv6 packet, given a typical MTU of 1500 bytes or smaller, is approximately 30.

- DHCPv6 prefix delegation [RFC3633]. DHCPv6 PD allows the client to request and be delegated a prefix, from which it can autonomously form other addresses. If the prefix is shorter than /64, it can be divided into multiple subnets which can be further delegated to downstream clients. If the prefix is a /64, it can be extended via L2 bridging, ND proxying [RFC4389] or /64 sharing [RFC7278], but it cannot be further subdivided, as a prefix longer than /64 is outside the current IPv6 specifications [RFC7421]. While [RFC3633] assumes that the DHCPv6 client is a router, DHCPv6 PD itself does not require that the client forward IPv6 packets not addressed to itself, and thus does not require that the client be an IPv6 router as defined in [RFC2460]. Also, in many cases (such as tethering, or hosting virtual machines), hosts are already forwarding IPv6 packets and thus operating as IPv6 routers as defined in [RFC2460].

	SLAAC	DHCPv6 IA_NA / IA_TA	DHCPv6 PD	DHCPv4
Can extend network	No+	No	Yes	Yes (NAT44)
Can number "unlimited" endpoints	Yes*	Yes*	No	No
Uses stateful, request-based assignment	No	Yes	Yes	Yes
Is immune to layer 3 on-link resource exhaustion attacks	No	Yes	Yes	Yes

[*] Subject to network limitations, e.g., ND cache entry size limits.
[+] Except on certain networks, e.g., [RFC7278].

Table 1: Comparison of multiple address assignment options

7. Number of addresses required

If we itemize the use cases from section Section 3, we can estimate the number of addresses currently used in normal operations. In typical implementations, privacy addresses use up to 8 addresses - one per day ([RFC4941] section 3.5). Current mobile devices may typically support 8 clients, with each one requiring one or more addresses. A client might choose to run several virtual machines.

Current implementations of 464XLAT require use of a separate address. Some devices require another address for their baseband chip. Even a host performing just a few of these functions simultaneously might need on the order of 20 addresses at the same time. Future applications designed to use an address per application or even per resource will require many more. These will not function on networks that enforce a hard limit on the number of addresses provided to hosts. Thus, in general is is not possible to estimate in advance how many addresses are required.

8. Recommendations

In order to avoid the problems described above, and preserve the Internet's ability to support new applications that use more than one IPv6 address, it is RECOMMENDED that IPv6 network deployments provide multiple IPv6 addresses from each prefix to general-purpose hosts. To support future use cases, it is NOT RECOMMENDED to impose a hard limit on the size of the address pool assigned to a host. Particularly, it is NOT RECOMMENDED to limit a host to only one IPv6 address per prefix.

Due to the drawbacks imposed by requiring explicit requests for address space (see section Section 4), it is RECOMMENDED that the network give the host the ability to use new addresses without requiring explicit requests. This can be achieved either by allowing the host to form new addresses autonomously (e.g., via SLAAC), or by providing the host with a dedicated /64 prefix. The prefix MAY be provided using DHCPv6 PD, SLAAC with per-device VLANs, or any other means.

Using stateful address assignment (DHCPv6 IA_NA or IA_TA) to provide multiple addresses when the host connects (e.g. the approximately 30 addresses that can fit into a single packet) would accommodate current clients, but sets a limit on the number of addresses available to hosts when they attach and would limit the development of future applications.

9. Operational considerations

9.1. Host tracking

Some network operators - often operators of networks that provide services to third parties such as university campus networks - are required to track which IP addresses are assigned to which hosts on their network. Maintaining persistent logs that map user IP addresses and timestamps to hardware identifiers such as MAC addresses may be used to attribute liability for copyright infringement or other illegal activity.

It is worth noting that this requirement can be met without using DHCPv6 address assignment. For example, it is possible to maintain these mappings by monitoring IPv6 neighbor table: routers typically allow periodic dumps of the neighbor cache via SNMP or other means, and many can be configured to log every change to the neighbor cache. Using SLAAC with a dedicated /64 prefix simplifies tracking, as it does not require logging each address formed by the host, but only the prefix assigned to the host when it attaches to the network. Similarly, providing address space using DHCPv6 PD has the same tracking properties as DHCPv6 address assignment, but allows the network to provide unrestricted address space.

Many large enterprise networks are fully dual-stack and implement address monitoring without using or supporting DHCPv6. The authors are directly aware of several networks that operate in this way, including the Universities of Loughborough, Minnesota, Reading, Southampton, Wisconsin and Imperial College London, in addition to the enterprise networks of the authors' employers.

It should also be noted that using DHCPv6 address assignment does not ensure that the network can reliably track the IPv6 addresses used by hosts. On any shared network without L2 edge port security, hosts are able to choose their own addresses regardless of what address provisioning methodology is in use. The only way to restrict the addresses used by hosts is to use layer 2 security mechanisms that enforce that particular IPv6 addresses are used by particular link-layer addresses (for example, SAVI [RFC7039]). If those mechanisms are available, it is possible to use them to provide tracking; this form of tracking is more secure and reliable than server logs because it operates independently of how addresses are allocated. Finally, tracking address information via DHCPv6 server logs is likely to become decreasingly viable due to ongoing efforts to improve the privacy of DHCPv6 and MAC address randomization [RFC7844].

9.2. Address space management

In IPv4, all but the world's largest networks can be addressed using private space [RFC1918], with each host receiving one IPv4 address. Many networks can be numbered in 192.168.0.0/16 which has roughly 64k addresses. In IPv6, that is equivalent to a /48, with each of 64k hosts receiving a /64 prefix. Under current RIR policies, a /48 is easy to obtain for an enterprise network. Networks that need a bigger block of private space use 10.0.0.0/8, which has roughly 16 million addresses. In IPv6, that is equivalent to a /40, with each host receiving /64 prefix. Enterprises of such size can easily obtain a /40 under current RIR policies.

In the above cases, aggregation and routing can be equivalent to IPv4: if a network aggregates per-host IPv4 addresses into prefixes of length /32 - n, it can aggregate per-host /64 prefixes into the same number of prefixes of length /64 - n.

Currently, residential users typically receive one IPv4 address and a /48, /56 or /60 IPv6 prefix. While such networks do not provide enough space to assign a /64 per host, such networks almost universally use SLAAC, and thus do not pose any particular limit to the number of addresses hosts can use.

Unlike IPv4 where addresses came at a premium, in all these networks, there is enough IPv6 address space to supply clients with multiple IPv6 addresses.

9.3. Addressing link layer scalability issues via IP routing

The number of IPv6 addresses on a link has direct impact for networking infrastructure nodes (routers, switches) and other nodes on the link. Setting aside exhaustion attacks via Layer 2 address spoofing, every (Layer 2, IP) address pair impacts networking hardware requirements in terms of memory, MLD snooping, solicited node multicast groups, etc. Many of these costs are incurred by neighboring hosts.

Hosts on such networks that create unreasonable numbers of addresses risk impairing network connectivity for themselves and other hosts on the network, and in extreme cases (e.g., hundreds or thousands of addresses) may even find their network access restricted by denial-of-service protection mechanisms.

We expect these scaling limitations to change over time as hardware and applications evolve. However, switching to a dedicated /64 prefix per host can resolve these scaling limitations. If the prefix is provided via DHCPv6 PD, or if the prefix can be used by only one link-layer address (e.g., if the link layer uniquely identifies or authenticates hosts based on MAC addresses), then there will be only one routing entry and one ND cache entry per host on the network. Furthermore, if the host is aware that the prefix is dedicated (e.g., if it was provided via DHCPv6 PD and not SLAAC), it is possible for the host to assign IPv6 addresses from this prefix to an internal interface such as a loopback interface. This obviates the need to perform Neighbor Discovery and Duplicate Address Detection on the network interface for these addresses, reducing network traffic.

Thus, assigning a dedicated /64 prefix per host is operationally prudent. Clearly, however, it requires more IPv6 address space than

using shared links, so the benefits provided must be weighed with the operational overhead of address space management.

10. Acknowledgements

The authors thank Tore Anderson, Brian Carpenter, David Farmer, Wesley George, Geoff Huston, Erik Kline, Victor Kuarsingh, Shucheng (Will) Liu, Dieter Siegmund, Mark Smith, Sander Steffann, Fred Templin and James Woodyatt for their input and contributions.

11. IANA Considerations

This memo includes no request to IANA.

12. Security Considerations

As mentioned in section 9.3, on shared networks using SLAAC it is possible for hosts to attempt to exhaust network resources and possibly deny service to other hosts by creating unreasonable numbers (e.g., hundreds or thousands) of addresses. Networks that provide access to untrusted hosts can mitigate this threat by providing a dedicated /64 prefix per host. It is also possible to mitigate the threat by limiting the number of ND cache entries that can be created for a particular host, but care must be taken to ensure that the network does not restrict the IP addresses available to non-malicious hosts.

Security issues related to host tracking are discussed in section 9.1.

13. References

13.1. Normative References

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

13.2. Informative References

[I-D.herbert-nvo3-ila] Herbert, T., "Identifier-locator addressing for network virtualization", draft-herbert-nvo3-ila-02 (work in progress), March 2016.

[I-D.tsvwg-quic-protocol]
Hamilton, R., Iyengar, J., Swett, I., and A. Wilk, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2", draft-tsvwg-quic-protocol-02 (work in progress), January 2016.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2993] Hain, T., "Architectural Implications of NAT", RFC 2993, DOI 10.17487/RFC2993, November 2000, <http://www.rfc-editor.org/info/rfc2993>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <http://www.rfc-editor.org/info/rfc3633>.

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

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <http://www.rfc-editor.org/info/rfc4389>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <http://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC5902] Thaler, D., Zhang, L., and G. Lebovitz, "IAB Thoughts on IPv6 Network Address Translation", RFC 5902, DOI 10.17487/RFC5902, July 2010, <http://www.rfc-editor.org/info/rfc5902>.

[RFC6434] Jankiewicz, E., Loughney, J., and T. Narten, "IPv6 Node Requirements", RFC 6434, DOI 10.17487/RFC6434, December 2011, <http://www.rfc-editor.org/info/rfc6434>.

[RFC6459] Korhonen, J., Ed., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, DOI 10.17487/RFC6459, January 2012, <http://www.rfc-editor.org/info/rfc6459>.

[RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", RFC 6877, DOI 10.17487/RFC6877, April 2013, <http://www.rfc-editor.org/info/rfc6877>.

[RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <http://www.rfc-editor.org/info/rfc7039>.

[RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <http://www.rfc-editor.org/info/rfc7217>.

[RFC7278] Byrne, C., Drown, D., and A. Vizdal, "Extending an IPv6 /64 Prefix from a Third Generation Partnership Project (3GPP) Mobile Interface to a LAN Link", RFC 7278, DOI 10.17487/RFC7278, June 2014, <http://www.rfc-editor.org/info/rfc7278>.

[RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <http://www.rfc-editor.org/info/rfc7421>.

[RFC7844] Huitema, C., Mrugalski, T., and S. Krishnan, "Anonymity Profiles for DHCP Clients", RFC 7844, DOI 10.17487/RFC7844, May 2016, <http://www.rfc-editor.org/info/rfc7844>.

[TARP] Gleitz, PM. and SM. Bellovin, "Transient Addressing for Related Processes: Improved Firewalling by Using IPv6 and Multiple Addresses per Host", August 2001.

[TS.24327] 3GPP, "Mobility between 3GPP Wireless Local Area Network (WLAN) interworking (I-WLAN) and 3GPP systems; General Packet Radio System (GPRS) and 3GPP I-WLAN aspects; Stage 3", June 2011.

Authors' Addresses

Lorenzo Colitti
Google
Roppongi 6-10-1
Minato, Tokyo 106-6126
JP

Email: lorenzo@google.com

Vint Cerf
Google
1875 Explorer St
10th Floor
Reston, VA 20190
US

Email: vint@google.com

Stuart Cheshire
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
US

Email: cheshire@apple.com

David Schinazi
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
US

Email: dschinazi@apple.com

IPv6 Operations Working Group (v6ops)
Internet-Draft
Intended status: Informational
Expires: June 12, 2016

F. Gont
SI6 Networks / UTN-FRH
J. Linkova
Google
T. Chown
University of Southampton
W. Liu
Huawei Technologies

December 10, 2015

Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World

draft-ietf-v6ops-ipv6-ehs-in-real-world-02

Abstract

This document presents real-world data regarding the extent to which packets with IPv6 extension headers are dropped in the Internet (as originally measured in August 2014 and later in June 2015, with similar results), and where in the network such dropping occurs. The aforementioned results serve as a problem statement that is expected to trigger operational advice on the filtering of IPv6 packets carrying IPv6 Extension Headers, so that the situation improves over time. This document also explains how the aforementioned results were obtained, such that the corresponding measurements can be reproduced by other members of the community and repeated over time to observe changes in the handling of packets with IPv6 extension headers.

Table of Contents

1. Introduction 2
2. Support of IPv6 Extension Headers in the Internet 3
3. IANA Considerations 6
4. Security Considerations 6
5. Acknowledgements 6
6. References 6
6.1. Normative References 6
6.2. Informative References 7
Appendix A. Reproducing Our Experiment 8
A.1. Obtaining the List of Domain Names 8
A.2. Obtaining AAAA Resource Records 9
A.3. Filtering the IPv6 Address Datasets 9
A.4. Performing Measurements with Each IPv6 Address Dataset . 10
A.5. Obtaining Statistics from our Measurements 11
Appendix B. Measurements Caveats 12
B.1. Isolating the Dropping Node 12
B.2. Obtaining the Responsible Organization for the Packet Drops 13
Appendix C. Troubleshooting Packet Drops due to IPv6 Extension Headers 14
Authors' Addresses 14

1. Introduction

IPv6 Extension Headers (EHs) allow for the extension of the IPv6 protocol, and provide support for core functionality such as IPv6 fragmentation. While packets employing IPv6 Extension Headers have been suspected to be dropped in some IPv6 deployments, there was not much concrete data on the topic. Some preliminary measurements have been presented in [PMTUD-Blackholes], [Gont-IEPG88] and

[Gont-Chown-IEPG89], whereas [Linkova-Gont-IEPG90] presents more comprehensive results on which this document is based.

This document presents real-world data regarding the extent to which packets containing IPv6 Extension Headers are dropped in the Internet, as measured in August 2014 and later in June 2015 with similar results (pending operational advice in this area). The results presented in this document indicate that in the scenarios where the corresponding measurements were performed, the use of IPv6 extension headers can lead to packet drops. We note that, in particular, packet drops occurring at transit networks are undesirable, and it is hoped and expected that this situation will improve over time.

2. Support of IPv6 Extension Headers in the Internet

This section summarizes the results obtained when measuring the support of IPv6 Extension Headers on the path towards different types of public IPv6 servers. Two sources of information were employed for the list of public IPv6 servers: the "World IPv6 Launch Day" site (http://www.worldipv6launch.org/) and Alexa's top 1M web sites (http://www.alexa.com). For each list of domain names, the following datasets were obtained:

- Web servers (AAAA records of the aforementioned list)
- Mail servers (MX -> AAAA of the aforementioned list)
- Name servers (NS -> AAAA of the aforementioned list)

Duplicate addresses and IPv6 addresses other than global unicast addresses were eliminated from each of those lists prior to obtaining the results included in this document. Additionally, addresses that were found to be unreachable were discarded from the dataset (please see Appendix B for further details).

For each of the aforementioned address sets, three different types of probes were employed:

- IPv6 packets with a Destination Options header of 8 bytes
- IPv6 packets resulting in two IPv6 fragments of 512 bytes each (approximately)
- IPv6 packets with a Hop-by-Hop Options header of 8 bytes

In the case of packets with a Destination Options Header and the case of packets with a Hop-by-Hop Options header, the desired EH size was

achieved by means of PadN options [RFC2460]. The upper-layer protocol of the probe packets was, in all cases, TCP [RFC0793] segments with the Destination Port set to the service port [IANA-PORT-NUMBERS] of the corresponding dataset. For example, the probe packets for all the measurements involving web servers were TCP segments with the destination port set to 80.

Besides obtaining the packet drop rate when employing the aforementioned IPv6 extension headers, we tried to identify whether the Autonomous System (AS) dropping the packets was the same as the Autonomous System of the destination/target address. This is of particular interest since it essentially reveals whether the packet drops are under the control of the intended destination of the packets. Packets dropped by the destination AS are less of a concern, since the device dropping the packets is under the control of the same organization as that to which the packets are destined (hence, it is probably easier to update the filtering policy if deemed necessary). On the other hand, packets dropped by transit ASes are more of a concern, since they affect the deployability and usability of IPv6 extension headers (including IPv6 fragmentation) by a third-party (the destination AS). In any case, we note that it is impossible to tell whether, in those cases where IPv6 packets with extension headers get dropped, the packet drops are the result of an explicit and intended policy, or the result of improper device configuration defaults, buggy devices, etc. Thus, packet drops that occur at the destination AS might still prove to be problematic.

Since there is some ambiguity when identifying the autonomous system to which a specific router belongs (see Appendix B.2), each of our measurements results in two different values: one corresponding to the "best-case scenario", and one corresponding to the "worst-case scenario". The "best-case scenario" is that in which, when in doubt, the packets are assumed to be dropped by the destination AS, whereas the "worst-case scenario" is that in which, when in doubt, the packets are assumed to be dropped by a transit AS (please see Appendix B.2 for details). In the following tables, the values shown within parentheses represent the possibility that, when a packet is dropped, the packet drop occurs in an AS other than the destination AS (considering both the best-case scenario and the worst-case scenario).

Dataset	DO8	HBH8	FH512
Webservers	11.88% (17.60%/20.80%)	40.70% (31.43%/40.00%)	30.51% (5.08%/6.78%)
Mailservers	17.07% (6.35%/26.98%)	48.86% (40.50%/65.42%)	39.17% (2.91%/12.73%)
Nameservers	15.37% (14.29%/33.46%)	43.25% (42.49%/72.07%)	38.55% (3.90%/13.96%)

Table 1: WIPv6LD dataset: Packet drop rate for different destination types, and estimated percentage of dropped packets that were deemed to be dropped in a different AS (lower, in parentheses)

NOTE: As an example, we note that the cell describing the support of IPv6 packets with DO8 for webservers (containing the value "11.88% (17.60%/20.80%)") should be read as: "when sending IPv6 packets with DO8 to public webservers, 11.88% of such packets get dropped. Among those packets that get dropped, 17.60%/20.80% (best case / worst case) of them get dropped at an AS other than the destination AS".

Dataset	DO8	HBH8	FH512
Webservers	10.91% (46.52%/53.23%)	39.03% (36.90%/46.35%)	28.26% (53.64%/61.43%)
Mailservers	11.54% (2.41%/21.08%)	45.45% (41.27%/61.13%)	35.68% (3.15%/10.92%)
Nameservers	21.33% (10.27%/56.80%)	54.12% (50.64%/81.00%)	55.23% (5.66%/32.23%)

Table 2: Alexa's top 1M sites dataset: Packet drop rate for different destination types, and estimated percentage of dropped packets that were deemed to be dropped in a different AS (lower, in parentheses)

There are a number of observations to be made based on the results presented above. Firstly, while it has been generally assumed that it is IPv6 fragments that are dropped by operators, our results indicate that it is IPv6 extension headers in general that result in packet drops. Secondly, our results indicate that a significant percentage of such packet drops occurs in transit Autonomous Systems;

that is, the packet drops are not under the control of the same organization as the final destination.

3. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

4. Security Considerations

This document presents real-world data regarding the extent to which IPv6 packets employing extension headers are dropped in the Internet. As such, this document does not introduce any new security issues.

5. Acknowledgements

The authors would like to thank (in alphabetical order) Mikael Abrahamsson, Mark Andrews, Fred Baker, Brian Carpenter, Gert Doering, C. M. Heard, Nick Hilliard, Joel Jaeggli, Tatuya Jinmei, Merike Kaeo, Warren Kumari, Ted Lemon, Mark Smith, Ole Troan, and Eric Vyncke, for providing valuable comments on earlier versions of this document. Additionally, the authors would like to thank participants of the v6ops and opsec working groups for their valuable input on the topics discussed in this document.

The authors would like to thank Fred Baker for his guidance in improving this document.

Fernando Gont would like to thank Jan Zorz / Go6 Lab <http://go6lab.si/>, and Jared Mauch / NTT America, for providing access to systems and networks that were employed to produce some of the measurement results presented in this document. Additionally, he would like to thank SixXS <https://www.sixxs.net> for providing IPv6 connectivity.

6. References

6.1. Normative References

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

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

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

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

6.2. Informative References

[blackhole6]
blackhole6, , "blackhole6 tool manual page", <http://www.si6networks.com/tools/ipv6toolkit>, 2014.

[Gont-Chown-IEPG89]
Gont, F. and T. Chown, "A Small Update on the Use of IPv6 Extension Headers", IEPG 89. London, UK. March 2, 2014, <http://www.iepg.org/2014-03-02-ietf89/fgont-iepg-ietf89-eh-update.pdf>.

[Gont-IEPG88]
Gont, F., "Fragmentation and Extension header Support in the IPv6 Internet", IEPG 88. Vancouver, BC, Canada. November 13, 2013, <http://www.iepg.org/2013-11-ietf88/fgont-iepg-ietf88-ipv6-frag-and-eh.pdf>.

[IANA-PORT-NUMBERS]
IANA, "Service Name and Transport Protocol Port Number Registry", <http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt>.

[IPv6-Toolkit]
"SI6 Networks' IPv6 Toolkit", <http://www.si6networks.com/tools/ipv6toolkit>.

[Linkova-Gont-IEPG90]
Linkova, J. and F. Gont, "IPv6 Extension Headers in the Real World v2.0", IEPG 90. Toronto, ON, Canada. July 20, 2014, <http://www.iepg.org/2014-07-20-ietf90/iepg-ietf90-ipv6-ehs-in-the-real-world-v2.0.pdf>.

[path6] path6, , "path6 tool manual page", <http://www.si6networks.com/tools/ipv6toolkit>, 2014.

[PMTUD-Blackholes] De Boer, M. and J. Bosma, "Discovering Path MTU black holes on the Internet using RIPE Atlas", July 2012, <http://www.nlnetlabs.nl/downloads/publications/pmtu-black-holes-msc-thesis.pdf>.

[RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, DOI 10.17487/RFC5927, July 2010, <http://www.rfc-editor.org/info/rfc5927>.

[RFC6980] Gont, F., "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery", RFC 6980, DOI 10.17487/RFC6980, August 2013, <http://www.rfc-editor.org/info/rfc6980>.

[RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <http://www.rfc-editor.org/info/rfc7045>.

[RFC7113] Gont, F., "Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)", RFC 7113, DOI 10.17487/RFC7113, February 2014, <http://www.rfc-editor.org/info/rfc7113>.

[RFC7123] Gont, F. and W. Liu, "Security Implications of IPv6 on IPv4 Networks", RFC 7123, DOI 10.17487/RFC7123, February 2014, <http://www.rfc-editor.org/info/rfc7123>.

Appendix A. Reproducing Our Experiment

This section describes, step by step, how to reproduce the experiment with which we obtained the results presented in this document. Each subsection represents one step in the experiment. The tools employed for the experiment are traditional UNIX-like tools (such as gunzip), and the SI6 Networks' IPv6 Toolkit [IPv6-Toolkit].

A.1. Obtaining the List of Domain Names

The primary data source employed was Alexa's Top 1M web sites, available at: <http://s3.amazonaws.com/alexa-static/top-1m.csv.zip>. The file is a zipped file containing the list of the most popular web sites, in CSV format. The aforementioned file can be extracted with "gunzip < top-1m.csv.zip > top-1m.csv".

A list of domain names (i.e., other data stripped) can be obtained with the following command of [IPv6-Toolkit]: "cat top-1m.csv | script6 get-alexa-domains > top-1m.txt". This command will create a "top-1m.txt" file, containing one domain name per line.

NOTE: The domain names corresponding to the WIPv6LD dataset is available at: <http://www.si6networks.com/datasets/wipv6day-domains.txt>. Since the corresponding file is a text file containing one domain name per line, the steps produced in this subsection need not be performed. The WIPv6LD data set should be processed in the same way as the Alexa Dataset, starting from Appendix A.2.

A.2. Obtaining AAAA Resource Records

The file obtained in the previous subsection contains a list of domain names that correspond to web sites. The AAAA records for such domain names can be obtained with:

```
$ cat top-1m.txt | script6 get-aaaa > top-1m-web-aaaa.txt
```

The AAAA records corresponding to the mailservers of each of the aforementioned domain names can be obtained with:

```
$ cat top-1m.txt | script6 get-mx | script6 get-aaaa > top-1m-mail-aaaa.txt
```

The AAAA records corresponding to the nameservers of each of the aforementioned domain names can be obtained with:

```
$ cat top-1m.txt | script6 get-ns | script6 get-aaaa > top-1m-dns-aaaa.txt
```

A.3. Filtering the IPv6 Address Datasets

The lists of IPv6 addresses obtained in the previous step could possibly contain undesired addresses (i.e., non-global unicast addresses) and/or duplicate addresses. In order to remove both undesired and duplicate addresses, each of the three files from the previous section should be filtered accordingly:

```
$ cat top-1m-web-aaaa.txt | addr6 -i -q -B multicast -B unspec -k
global > top-1m-web-aaaa-unique.txt

$ cat top-1m-mail-aaaa.txt | addr6 -i -q -B multicast -B unspec -k
global > top-1m-mail-aaaa-unique.txt
```

```
$ cat top-1m-dns-aaaa.txt | addr6 -i -q -B multicast -B unspec -k
global > top-1m-dns-aaaa-unique.txt
```

A.4. Performing Measurements with Each IPv6 Address Dataset

A.4.1. Measurements with web servers

In order to measure DO8 with the list of webservers:

```
# cat top-1m-web-aaaa-unique.txt | script6 trace6 do8 tcp 80 > top-
1m-web-aaaa-do8-m.txt
```

In order to measure HBH8 with the list of webservers:

```
# cat top-1m-web-aaaa-unique.txt | script6 trace6 hbh8 tcp 80 > top-
1m-web-aaaa-hbh8-m.txt
```

In order to measure FH512 with the list of webservers:

```
# cat top-1m-web-aaaa-unique.txt | script6 trace6 fh512 tcp 80 > top-
1m-web-aaaa-fh512-m.txt
```

A.4.2. Measurements with mail servers

In order to measure DO8 with the list of mailservers:

```
# cat top-1m-mail-aaaa-unique.txt | script6 trace6 do8 tcp 25 > top-
1m-mail-aaaa-do8-m.txt
```

In order to measure HBH8 with the list of webservers:

```
# cat top-1m-mail-aaaa-unique.txt | script6 trace6 hbh8 tcp 25 > top-
1m-mail-aaaa-hbh8-m.txt
```

In order to measure FH512 with the list of webservers:

```
# cat top-1m-mail-aaaa-unique.txt | script6 trace6 fh512 tcp 25 >
top-1m-mail-aaaa-fh512-m.txt
```

A.4.3. Measurements with DNS servers

In order to measure DO8 with the list of mameservers:

```
# cat top-1m-dns-aaaa-unique.txt | script6 trace6 do8 tcp 53 > top-
1m-dns-aaaa-do8-m.txt
```

In order to measure HBH8 with the list of webservers:

```
# cat top-1m-dns-aaaa-unique.txt | script6 trace6 hbh8 tcp 53 > top-
1m-dns-aaaa-hbh8-m.txt
```

In order to measure FH512 with the list of webservers:

```
# cat top-1m-dns-aaaa-unique.txt | script6 trace6 fh512 tcp 53 > top-
1m-dns-aaaa-fh512-m.txt
```

A.5. Obtaining Statistics from our Measurements

A.5.1. Statistics for Web Servers

In order to compute the statistics corresponding to our measurements of DO8 with the list of webservers:

```
$ cat top-1m-web-aaaa-do8-m.txt | script6 get-trace6-stats > top-1m-
web-aaaa-do8-stats.txt
```

In order to compute the statistics corresponding to our measurements of HBH8 with the list of webservers:

```
$ cat top-1m-web-aaaa-hbh8-m.txt | script6 get-trace6-stats > top-1m-
web-aaaa-hbh8-stats.txt
```

In order to compute the statistics corresponding to our measurements of FH512 with the list of webservers:

```
$ cat top-1m-web-aaaa-fh512-m.txt | script6 get-trace6-stats > top-
1m-web-aaaa-fh512-stats.txt
```

A.5.2. Statistics for Mail Servers

In order to compute the statistics corresponding to our measurements of DO8 with the list of mailservers:

```
$ cat top-1m-mail-aaaa-do8-m.txt | script6 get-trace6-stats > top-1m-
mail-aaaa-do8-stats.txt
```

In order to compute the statistics corresponding to our measurements of HBH8 with the list of mailservers:

```
$ cat top-1m-mail-aaaa-hbh8-m.txt | script6 get-trace6-stats > top-
1m-mail-aaaa-hbh8-stats.txt
```

In order to compute the statistics corresponding to our measurements of FH512 with the list of mailservers:

```
$ cat top-1m-mail-aaaa-fh512-m.txt | script6 get-trace6-stats > top-
1m-mail-aaaa-fh512-stats.txt
```

A.5.3. Statistics for Name Servers

In order to compute the statistics corresponding to our measurements of DO8 with the list of nameservers:

```
$ cat top-1m-dns-aaaa-do8-m.txt | script6 get-trace6-stats > top-1m-
dns-aaaa-do8-stats.txt
```

In order to compute the statistics corresponding to our measurements of HBH8 with the list of mailservers:

```
$ cat top-1m-dns-aaaa-hbh8-m.txt | script6 get-trace6-stats > top-1m-
dns-aaaa-hbh8-stats.txt
```

In order to compute the statistics corresponding to our measurements of FH512 with the list of mailservers:

```
$ cat top-1m-dns-aaaa-fh512-m.txt | script6 get-trace6-stats > top-
1m-dns-aaaa-fh512-stats.txt
```

Appendix B. Measurements Caveats

A number of issues have needed some consideration when producing the results presented in this document. These same issues should be considered when troubleshooting connectivity problems resulting from the use of IPv6 Extension headers.

B.1. Isolating the Dropping Node

Let us assume that we find that IPv6 packets with EHs are being dropped on their way to the destination system 2001:db8:d::1, and that the output of running traceroute towards such destination is:

1. 2001:db8:1:1000::1
2. 2001:db8:2:4000::1
3. 2001:db8:3:4000::1
4. 2001:db8:3:1000::1
5. 2001:db8:4:4000::1
6. 2001:db8:4:1000::1
7. 2001:db8:5:5000::1
8. 2001:db8:5:6000::1
9. 2001:db8:d::1

Additionally, let us assume that the output of EH-enabled traceroute to the same destination is:

1. 2001:db8:1:1000::1
2. 2001:db8:2:4000::1
3. 2001:db8:3:4000::1
4. 2001:db8:3:1000::1
5. 2001:db8:4:4000::1

For the sake of brevity, let us refer to the last-responding node in the EH-enabled traceroute ("2001:db8:4:4000::1" in this case) as "M". Assuming that packets in both traceroutes employ the same path, we'll refer to "the node following the last responding node in the EH-enabled traceroute" ("2001:db8:4:1000::1" in our case), as "M+1", etc.

Based on traceroute information above, which node is the one actually dropping the EH-enabled packets will depend on whether the dropping node filters packets before making the forwarding decision, or after making the forwarding decision. If the former, the dropping node will be M+1. If the latter, the dropping node will be "M".

Throughout this document (and our measurements), we assume that those nodes dropping packets that carry IPv6 EHs apply their filtering policy, and only then, if necessary, forward the packets. Thus, in our example above the last responding node to the EH-enabled traceroute ("M") is "2001:db8:4:4000::1", and therefore we assume the dropping node to be "2001:db8:4:1000::1" ("M+1").

Additionally, we note that when isolating the dropping node we assume that both the EH-enabled and the EH-free traceroutes result in the same paths. However, this might not be the case.

B.2. Obtaining the Responsible Organization for the Packet Drops

In order to identify the organization operating the dropping node, one would be tempted to lookup the ASN corresponding to the dropping node. However, assuming that M and M+1 are two peering routers, any of these two organizations could be providing the address space employed for such peering. Or, in the case of an Internet eXchange Point (IXP), the address space could correspond to the IXP AS, rather than to any of the participating ASes. Thus, the organization operating the dropping node (M+1) could be the AS for M+1, but it might as well be the AS for M+2. Only when the ASN for M+1 is the same as the ASN for M+2 we have certainty about who the responsible organization for the packet drops is (see slides 21-23 of [Linkova-Gont-IEPG90]).

In the measurement results presented in Section 2, the aforementioned ambiguity results in a "best-case" and a "worst-case" scenario (rather than a single value): the lowest percentage value means that,

when in doubt, we assume the packet drops occur in the same AS as the destination; on the other hand, the highest percentage value means that, when in doubt, we assume the packet drops occur at different AS than the destination AS.

We note that the aforementioned ambiguity should also be considered when troubleshooting and reporting IPv6 packet drops, since identifying the organization responsible for the packet drops might probe to be a non-trivial task.

Finally, we note that a specific organization might be operating more than one Autonomous System. However, our measurements assume that different Autonomous System Numbers imply different organizations.

Appendix C. Troubleshooting Packet Drops due to IPv6 Extension Headers

Isolating IPv6 blackholes essentially involves performing IPv6 traceroute for a destination system with and without IPv6 extension headers. The EH-free traceroute would provide the full working path towards a destination, while the EH-enabled traceroute would provide the address of the last-responding node for EH-enabled packets (say, "M"). In principle, one could isolate the dropping node by looking-up "M" in the EH-free traceroute, with the dropping node being "M+1" (see Appendix B.1 for caveats).

At the time of this writing, most traceroute implementations do not support IPv6 extension headers. However, the path6 tool [path6] of [IPv6-Toolkit] provides such support. Additionally, the blackhole6 tool [blackhole6] of [IPv6-Toolkit] automates the troubleshooting process and can readily provide information such as: dropping node's IPv6 address, dropping node's Autonomous System, etc.

Authors' Addresses

Fernando Gont
SI6 Networks / UTN-FRH
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

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

J. Linkova
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
USA

Email: furry@google.com

Tim Chown
University of Southampton
Highfield
Southampton , Hampshire SO17 1BJ
United Kingdom

Email: tjc@ecs.soton.ac.uk

Will(Shucheng) Liu
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: liushucheng@huawei.com

v6ops
Internet-Draft
Intended status: Best Current Practice
Expires: April 21, 2016

J. Brzozowski
Comcast Cable
G. Van De Velde
Alcatel-Lucent

October 19, 2015

Unique IPv6 Prefix Per Host
draft-jjmb-v6ops-unique-ipv6-prefix-per-host-00

Abstract

In some IPv6 environments the need has arisen for hosts to be able to utilise a unique IPv6 prefix even though the link or media may be shared. Typically hosts (subscribers) on a shared network, like Wi-Fi or Ethernet, will acquire unique IPv6 addresses from a common IPv6 prefix that is allocated or assigned for use on a specific link. Benefits of a unique IPv6 prefix compared to a unique IPv6 address from the service provider are going from enhanced subscriber management to improved isolation between subscribers.

In most deployments today IPv6 address assignment from a single IPv6 prefix on a shared network is done by either using IPv6 stateless address auto-configuration (SLAAC) and/or stateful DHCPv6. While this is still viable and operates as designed there are some large scale environments where this concept introduces significant performance challenges and implications, specifically related to IPv6 router and neighbor discovery. This document outlines an approach utilising existing IPv6 protocols to allow hosts to be assigned a unique IPv6 prefix (instead of a unique IPv6 address from a shared IPv6 prefix).

Table of Contents

1. Introduction 2
2. Motivation and Scope of Applicability 3
3. Design Princinples 4
4. Behaviour 4
4.1. Community Wi-Fi Network Topology Description 4
4.2. Wi-Fi Subscriber Onboarding Procedures 6
4.3. UE IPv6 Addressing and Configuration 10
4.3.1. IPv6 Addressing 10
4.3.2. IPv6 Configuration 11
5. Operational Considerations 11
6. Future work 13
7. IANA Considerations 13
8. Security Considerations 13
9. Acknowledgements 13
10. References 13
10.1. Normative References 13
10.2. Informative References 14
Authors' Addresses 14

1. Introduction

The concepts in this document were originally developed as part of a large scale, production deployment of IPv6 support for a community Wi-Fi service. In this document IPv6 support does not preclude support for IPv4, however, the primary objectives for this work was to make it so that user equipment (UE) were capable of an IPv6 only experience from a network operators perspective. Details of IPv4 support are out of scope for this document. This document will also, in general, outline the requirements that must be satified by UE to allow for an IPv6 only experience.

In most deployments today User Equipment (UE) IPv6 address assignment is commonly done using either IPv6 SLAAC RFC4862 [RFC4862] and/or DHCP IA_NA RFC3315 [RFC3315]. However, at current time there is a non-trivial UE/subscriber base not supporting DHCPv6 IA_NA, making IPv6 SLAAC based subscriber and address management for community Wi-Fi services the technology of choice as it does not exclude any known IPv6 implementation. This document will detail the mechanics involved for IPv6 SLAAC based address and subscriber management coupled with stateless DHCPv6, where beneficial.

A community Wi-Fi service is an environment to allow subscribers (hosts) to connect to a shared network providing Internet and/or closed network services. Often Service providers use community Wi-Fi networks to provide enhanced subscriber connectivity experiences. Additionally retail owners frequently provide community Wi-Fi services to improve their customers retail experience.

Upon further exploration the approach documented here has applicability in other environments including corporate, enterprise, or university settings where IPv6 support is desired over a shared media. Where applicable details related to the same will be provided.

2. Motivation and Scope of Applicability

The motivation for this work falls into the following categories:

- Deploy support for IPv6 that will allow for an IPv6 only experience, even if IPv4 support is present
- Ensure support for IPv6 is efficient and does not impact the performance of the underlying network and in turn the customer experience
- Allow for the greatest flexibility across host implementation to allow for the widest range of addressing and configuration mechanisms to be employed. The goal here is the ensure that the widest population of UE implementations can leverage the availability of IPv6.
- Lay the technological foundation for future work related to the use of IPv6 over shared media like Wi-Fi

While this work was originally conceived in the context of large scale Wi-Fi networks, the scope of applicability is much broader. The techniques and concepts or subsets of the same may also be applicable in residential or SOHO networking environments.

3. Design Princinples

The Wireless LAN Gateway (WLAN-GW) discussed in this document is the L3-Edge router responsible for the communication with the Wi-Fi subscribers (hosts) and to aggregate the traffic from the Wi-Fi subscribers and the Wireless LAN network towards the community Wi-Fi provider.

The goal of a WLAN-GW is to provide sufficient data-plane throughput capacity to aggregate all Wi-Fi subscriber traffic, while at the same time it is functioning as control-plane anchor point to make sure that each subscriber is receiving the expected subscriber policy and service levels (throughput, QoS, security, parental-control, subscriber mobility management, etc.).

The work detailed in this document intends to provide details regarding the WLAN-GW Wi-Fi subscriber/host addressing methodology. Evolved WLAN-GW capabilities regarding fixed/mobile convergence, traffic steering, etc. are not the main focus and are outside the scope of this document.

4. Behaviour

This section outlines the essential components of the described system and interaction amongst the same.

4.1. Community Wi-Fi Network Topology Description

The topology and design referenced in this document is a generalized description of functional components currently deployed in a large scale subscriber oriented network.

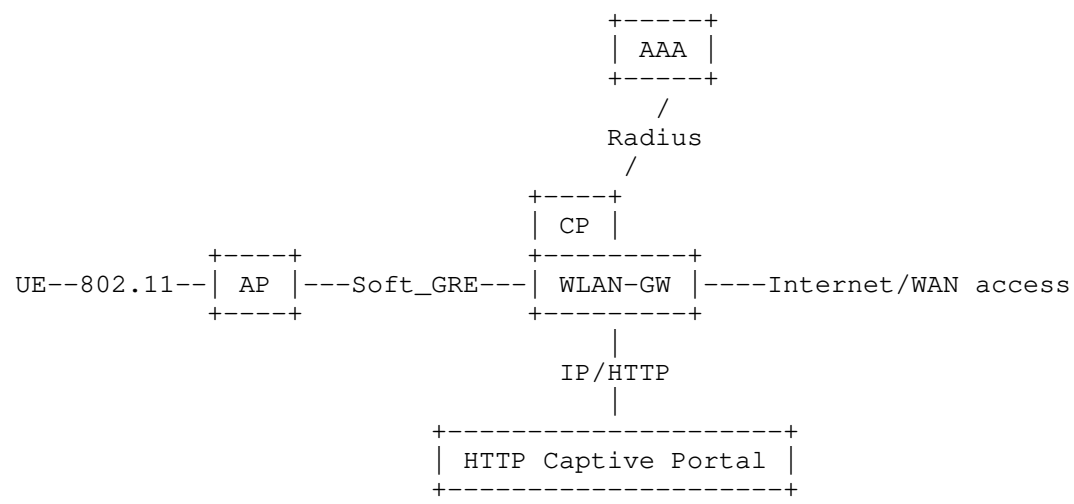


Figure 1

- UE: User Equipment.
- 802.11: Wireless Network
- AP: Access Point.
- Soft-GRE: Stateless GRE tunnel
- WLAN-GW: Wireless LAN Gateway
- CP: Control Plane component of the WLAN-GW
- AAA: Accounting, Authorisation and Authentication
- HTTP Captive Portal: Captive portal used to redirect traffic towards during subscriber onboarding process

While there are many ways for UE to associate to a Wi-Fi network (e.g. EAP-SIM, EAP-AKA, WPA2-PSK, etc.), community Wi-Fi predominantly leverages an HTTP Captive Portal. The key function for the Captive Portal is to identify the UE/subscriber and create on the WLAN-GW the corresponding UE/subscriber context for policy and accounting.

The Soft-GRE session is a stateless GRE tunnel between AP and the WLAN-GW. The AP is configured with the IP address or FQDN of the tunnel concentrator or aggregation point and initiates the GRE

tunnel, over IPv6 preferrably, by encapsulating packets towards the WLAN-GW. The WLAN-GW is configured as a GRE tunnel head-end server and accepts these GRE packets, while at the same time creating correct tunnel context to identify the AP. Soft-GRE is a very well established pragmatic technology. The use of GRE over IPv4 only furthers an operators dependence on IPv4 and should be deprecated by using GRE over IPv6 only.

The AP has, as seen in the illustration, an interface attached to the Wi-Fi network and will bridge traffic received on this Wi-Fi interface over the Soft-GRE tunnel to the WLAN-GW. This will include traffic from newly attached UE/subscribers which have not been identified or authorized on the Wi-Fi network. At the same time the AP implements split-horizon for BUM (broadcast, unknown and multicast) traffic, making sure that there is no undesired leakage of traffic between UE/subscribers attached to the Wi-Fi network.

The Control Plane (CP) of the WLAN-GW is a key component used during onboarding of UE/subscribers to identify the UE/subscriber and to exchange IP address related details. For that purpose it can make usage of DHCP, ARP, DHCPv6, ICMPv6 (RS/RA/NS/NA), Radius, Diameter, etc.

4.2. Wi-Fi Subscriber Onboarding Procedures

This section provides detail about Best Practice operational steps to onboard a UE/subscriber and the key architectural technology used to create the WLAN-GW UE/subscriber policy and IP addressing context.

The flow chart pictured below is providing a sequential overview of the operational steps performed to onboard a UE onto a community Wi-Fi network.

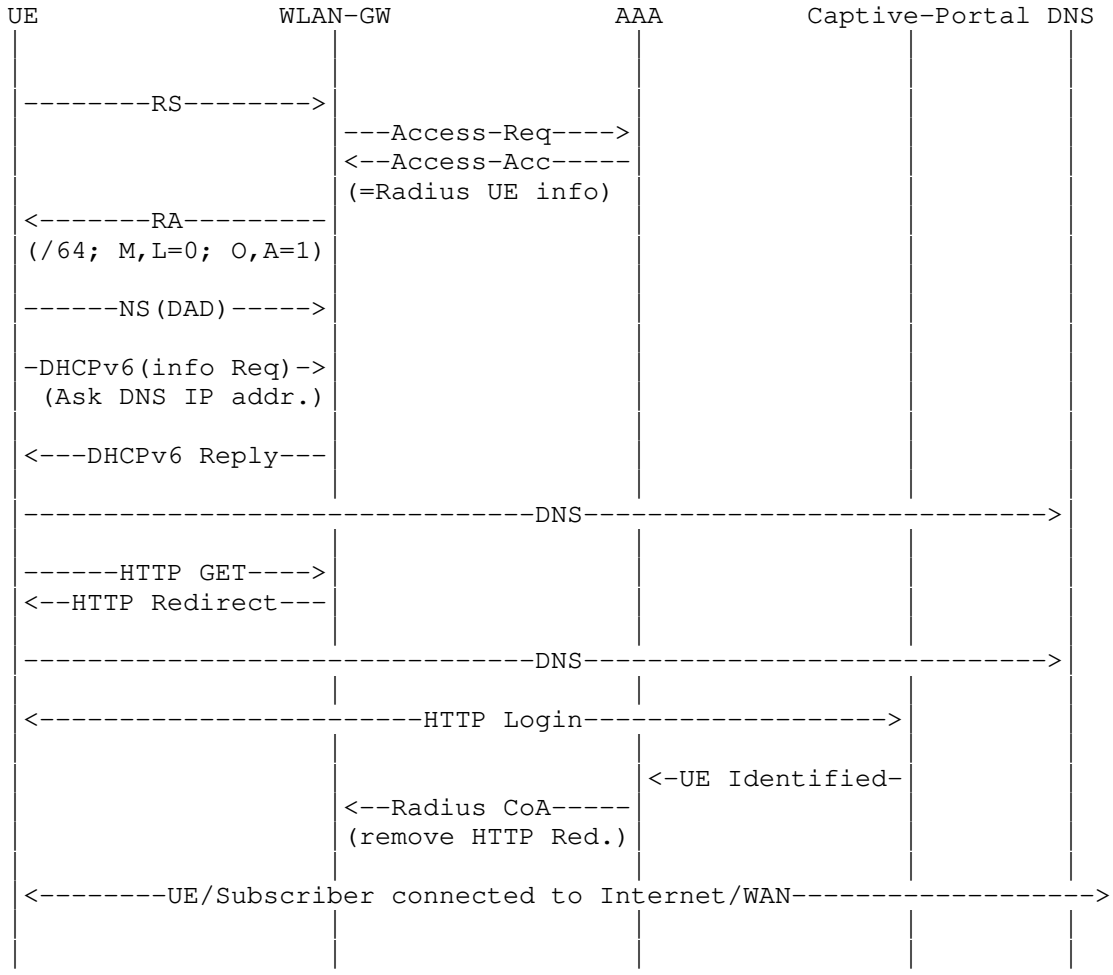


Figure 2

Note that the Wireless Access Point (AP) is not pictured in the flow chart above. This is because the AP is from architectural perspective functioning as a L2 bridge between the UE and WLAN-GW. For Wi-Fi community service the AP is configured to setup a Soft-GRE tunnel towards the WLAN-GW and to bridge relevant Wi-Fi traffic upon the Soft-GRE tunnel. The AP is also configured for split-horizon towards the Wi-Fi interface for subscriber isolation and security purpose. The AP will for the remainder of this document be silently inserted between UE and WLAN-GW to bridge traffic between the WLAN-GW and AP and vice versa

When a new UE connects to the community Wi-Fi it connects to the Wi-Fi network by attaching to the relevant 'open' SSID advertised for use as part of the community Wi-Fi offering. Once the UE/subscriber is attached to the Wi-Fi SSID it will initiate IP configuration. The focus of this document is to share IPv6 address assignment Best Practices, and hence will focus around those topics, eventhough there are many more aspects to deploy a quality community Wi-Fi service offering successfully.

Once the UE is connected to the Wi-Fi shared network, it will from an IPv6 perspective attempt to learn the default IPv6 gateway, the IPv6 prefix information, the DNS information, and the remaining information required to establish globally routable IPv6 connectivity. For that purpose the the UE/subscriber sends a RS (Router Solicitation) message. This RS is forwarded by the AP-bridge over the Soft-GRE interface, however due to the split-horizon configuration for BUM traffic it is not relayed to any other UE/ Subscribers attached to the Wi-Fi network.

The WLAN-GW received this UE/subscriber RS message, and because it is the first time this UE/subscriber attaches to the Wi-Fi the UE/ subscriber is by default not authorized. The WLAN-GW will now try to discover additional information about the subscriber information by querying the AAA server. This is done by sending a Radius Access-Request.

The Radius server receives this Access-Request, and performs a lookup in its policy database. If radius server discovers that the UE/ subscriber is a fresh device trying to gain access onto the Wi-Fi network it will identify some parameters (e.g. IPv6 /64 prefix) to send back to the WLAN-GW together with a message to install a HTTP-redirect to a Captive Portal for further UE/subscriber identification. This will be sent from the AAA server to the WLAN-GW in a Radius Access-Ackowledge message.

The WLAN-GW will use the received Radius information to compose the response to the UE/subscriber originated RS message. The WLAN-GW will answer using a unicast RA (Router Advertisement) to the UE/ subscriber. This RA contains a few important parameters for the EU/ subscriber to consume: (1) a /64 prefix and (2) flags. The /64 prefix can be derived from a locally managed pool or aggregate IPv6 block assigned to the WLAN-GW or from a pool signalled by the Radius server in a radius attribute. The flags may indicate to the UE/ subscriber to use SLAAC and/or DHCPv6 for address assignment, it may indicate if the autoconfigured address is on/off-link and if 'Other' information (e.g. DNS server address) needs to be requested.

The IPv6 RA flags used for best common practice in IPv6 SLAAC based community Wi-Fi are:

- M-flag = 0 (UE/subscriber address is not managed through DHCPv6), this flag may be set to 1 in the future if/when DHCPv6 prefix delegation support over Wi-Fi is desired)
- O-flag = 1 (DHCPv6 is used to request configuration information i.e. DNS, NTP information, not for IPv6 addressing)
- A-flag = 1 (The UE/subscriber can configure itself using SLAAC)
- L-flag = 0 (The UE/subscriber is off-link, which means that the UE/subscriber will send packets ALWAYS to his default gateway, even if the destination is within the range of the /64 prefix)

The use of a unique IPv6 prefix per UE adds an additional level of protection and efficiency as it relates to how IPv6 Neighbor Discovery and Router Discovery processing. Since the UE has a unique IPv6 prefix all traffic by default will be directed to the WLAN-GW. Further, the flag combinations documented above maximize the IPv6 configurations that are available by hosts including the use of privacy IPv6 addressing.

The architected result of designing the RA as documented above is that each UE/subscriber gets its own unique /64 IPv6 prefix for which it can use SLAAC or any other method to select its /128 unique address. In addition it will use stateless DHCPv6 to get the IPv6 address of the DNS server, however it SHOULD NOT use stateful DHCPv6 to receive a service provider managed IPv6 address. If the UE/ subscriber desires to send anything external including other UE/ subscriber devices (assuming device to device communications is enabled and supported), then due to the L-bit set it SHOULD send this traffic to the WLAN-GW.

Now that the UE/subscriber received the RA and the associated flags, it will assign itself a 128 bit IPv6 address using SLAAC. Since the address is composed by the UE/subscriber device itself it will need to verify that the address is unique on the shared network. The UE/ subscriber will for that purpose perform Duplicate Address Detection algorithm. This will occur for each address the UE attempts to utilize on the Wi-Fi network.

At this stage the UE/subscriber has acquired a valid IPv6 address, however it may not have received one or more DNS server IPv6 address. The UE/subscriber can use stateless DHCPv6 exchange to identify a valid DNS server address(es). An alternative solution, albeit less supported by IPv6 hosts is to signal DNS server addresses is by

utilising RA extensions described in RNDSS RFC6106 [RFC6106] in which the router uses Router Advertisement options to advertise a list of DNS recursive server addresses and a DNS Search List to IPv6 UE/ subscribers. The use of RNDSS and stateless DHCPv6 for the configuration of hosts are not mutually exclusive. Both methods can and should be enabled simultaneously allowing for the widest range of hosts or UEs to learn and use DNS over IPv6. DNS server IPv6 address(es) sent via DHCPv6 and RDNSS must be identical.

At this moment the UE/subscriber has all information to be connected to the Internet, nevertheless the community Wi-Fi service provider has no idea about the identity or credentials of the UE/subscriber. For that purpose the Service provider has installed on the WLAN-GW a HTTP redirect for this particular UE/subscriber towards HTTP captive portal. First the subscriber utilises DNS to correlate the domain name with an IP address, next the HTTP GET is intercepted and an HTTP Redirect is issued to Redirect the HTTP session towards the Captive portal. The ultimate goal of this process is for the service provider to identify the UE/subscriber. From the moment the UE/ subscriber identified itself on the captive portal (login-ID/PW, PIN Challenge, etc.) then the captive portal informs the WLAN-GW about the correct policies (QoS, policing, etc.) and to remove the HTTP-redirect.

From now onwards the WLAN-GW has identified the UE/subscriber and installed all the subscriber context for identification, billing, traffic conditioning. The UE/subscriber can access the Internet/WAN within his agreed commuity Wi-Fi parameters.

4.3. UE IPv6 Addressing and Configuration

An over arching objective for any IPv6 deployment where subscriber endpoints or UEs are concerned must include an IPv6 only experience. Specifically, similar to residential broadband networks, Wi-Fi networks that support IPv6 must ensure there are no dependencies on IPv4. Due to fragmented support for various IPv6 address and configuration mechanisms network operators must effectively enable and support every combination of IPv6 address and configuration technique. Coordinating the configuraiton and values for the same is important to ensure proper UE behavior.

4.3.1. IPv6 Addressing

Stateless IPv6 address autoconfiguration is expected to be the primary mechanism for UEs to leverage when establing globally routable IPv6 connectivity. Stateful DHCPv6 is currently not utilized in this model for host addressing since stateful DHCPv6 is not universally supported for address acquisition. Stateful DHCPv6

may be considering in the future as part of enabling support for IPv6 prefix delegation [RFC3633].

4.3.2. IPv6 Configuration

In order to make an IPv6 only experience possible Wi-Fi network operators must ensure that UEs are able to reach all critical network services over IPv6. Today, many host operating systems still prefer querying DNS over IPv4. Additionally, widely deployed hosts do not truly leverage a single common approach for IPv6 configuration. As such the following should be expected to be available to support a proper IPv6 only configuration:

- RDNSS [RFC6106] is enabled by default and is expected to contain one or more globally routable IPv6 addresses
- Stateless DHCPv6 [RFC3315] is enabled by default and will minimally transmit one or more DNS server IPv6 addresses. To ensure the desired behavior is triggered IPv6 router advertisements transmitted by the WLAN-GW will set the M flag to 0 and the O flag to 1.

5. Operational Considerations

An operational consideration when using IPv6 address assignment using IPv6 SLAAC is that after the onboarding procedure the UE/subscriber will have a prefix with certain preferred and valid lifetimes. The WLAN-GW extends these lifetimes by sending an unsolicited RA, the applicable MaxRtrAdvInterval on the WLAN-GW MUST therefore be lower than the preferred lifetime. As a consequence of this process is that the WLAN-GW never knows when a UE/subscriber stops using addresses from a prefix and additional procedures are required to help the WLAN-GW to gain this information. When using stateful DHCPv6 IA_NA for IPv6 UE/subscriber address assignment this uncertainty on the WLAN-GW is not of impact due to the stateful nature of DHCPv6 IA_NA address assignment.

Following is reference table of the key IPv6 router discovery and neighbor discovery timers:

- IPv6 Router Advertisement Interval = 300s
- IPv6 Router LifeTime = 3600s
- Reachable time = 30s
- IPv6 Valid Lifetime = 3600s

- IPv6 Preferred Lifetime = 1800s
- Retransmit timer = 0s

The stateless nature of the UE/subscriber IPv6 SLAAC connectivity model provides value to make sure that the UE/subscriber context is timely removed from the WLAN-GW to avoid ongoing resource depletion. A possible solution is to use a subscriber inactivity timer which after tracking a pre-defined (currently unspecified) # of minutes deletes the subscriber context on the WLAN-GW.

When using SLAAC the UE/subscriber the IP address assignment happens without a WLAN-GW controlled state machine, and as result there is no state-information on the WLAN-GW about actual IPv6 address usage. To accomodate this the WLAN-GW can periodically perform a Subscriber Host Connectivity Verification (i.e. periodically ping each IPv6 UE/subscriber from the WLAN-GW) to make sure that the subscriber table on the WLAN-GW is correct and that the inactive UE/subscribers are removed.

When employing stateless IPv6 address assignment a number of widely deployed operating systems will attempt to utilize RFC 4941 RFC4941 [RFC4941] temporary 'private' addresses. This can lead to the consequence that a UE has multiple /128 addresses from the same IPv6 prefix. The WLAN-GW MUST be able to handle the presence and use of multiple globally routable IPv6 addresses.

When geo-localisation is of importance the WLAN-GW needs to have information about the Access Point to which the UE/subscriber is connected. In an environment using DHCPv6 IA_NA for IPv6 address assignment this is achieved by having the AP insert an interface-id RFC3315 [RFC3315] in the UE/subscriber DHCPv6 Solicit message. The interface-id format expected is [ap-mac;ssid;[o-s]], e.g. [00:11:22:33:44:55;example;o] (o stands for open, s for secure). This way the service provider can learn both the AP-MAC (identifies location) and the SSID (identifies service). When a service provider uses SLAAC IPv6 address assignment it becomes harder for the service provider to rely on this type of information and alternate solutions have to be used to acquire the MAC address of the Access Point to which the UE/subscriber is connected. A solution could be for the WLAN-GW to support NSoGRE to harvest the Access-Point MAC address to which the UE/subscriber is connected.

For security purposes it will be important for the service provider to have the capability on the WLAN-GW to have supported mechanics for LI (Lawfull Intercept) and the installation of IPv6 filters per subscriber.

For accounting purposes the WLAN-GW must be able to send usage statistics per UE/subscriber using Radius attributes.

6. Future work

- Support for IPv6 prefix delegation over Wi-Fi

7. IANA Considerations

No IANA considerations are defined at this time.

8. Security Considerations

No Additional Security Considerations are made in this document.

9. Acknowledgements

The authors would like to thank, in alphabetical order, for their contributions:

Lorenzo Colitti, Killian Desmedt, Brad Hilgenfeld, Wim Henderickx, Erik Kline, Thomas Lynn, Phil Sanderson, Colleen Szymanik, Sanjay Wadhwa

10. References

10.1. Normative References

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <http://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, DOI 10.17487/RFC6106, November 2010, <http://www.rfc-editor.org/info/rfc6106>.

[RFC6180] Arkko, J. and F. Baker, "Guidelines for Using IPv6 Transition Mechanisms during IPv6 Deployment", RFC 6180, DOI 10.17487/RFC6180, May 2011, <http://www.rfc-editor.org/info/rfc6180>.

10.2. Informative References

[I-D.ietf-v6ops-v4v6tran-framework] Carpenter, B., Jiang, S., and V. Kuarsingh, "Framework for IP Version Transition Scenarios", draft-ietf-v6ops-v4v6tran-framework-02 (work in progress), July 2011.

[RFC6343] Carpenter, B., "Advisory Guidelines for 6to4 Deployment", RFC 6343, DOI 10.17487/RFC6343, August 2011, <http://www.rfc-editor.org/info/rfc6343>.

Authors' Addresses

John Jason Brzozowski
Comcast Cable
1701 John F. Kennedy Blvd.
Philadelphia, PA
USA

Email: john_brzozowski@cable.comcast.com

Gunter Van De Velde
Alcatel-Lucent
Antwerp
Belgium

Email: gunter.van_de_velde@alcatel-lucent.com