Explicit Address Mappings for Stateless IP/ICMP Translation
draft-anderson-v6ops-siit-eam-03

Abstract

This document extends the Stateless IP/ICMP Translation Algorithm (SIIT) with an Explicit Address Mapping (EAM) algorithm, and formally updates RFC 6145. The EAM algorithm facilitates stateless IP/ICMP translation between arbitrary (non-IPv4-translatable) IPv6 endpoints and IPv4.

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 July 12, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.
1.  Introduction  . . . . . . . . . . . . . . . . . . . . . . .   2
  1.1.  Terminology . . . . . . . . . . . . . . . . . . . . . . .   3
2.  Problem Statement . . . . . . . . . . . . . . . . . . . . . .   3
3.  Explicit Address Mapping Algorithm  . . . . . . . . . . . . .   5
  3.1.  Explicit Address Mapping Table  . . . . . . . . . . . . .   5
  3.2.  Explicit Address Mapping Specification  . . . . . . . . .   6
  3.3.  IP Address Translation Procedure  . . . . . . . . . . . . .   6
    3.3.1.  Address Translation Steps: IPv4 to IPv6  . . . . . . .   7
    3.3.2.  Address Translation Steps: IPv6 to IPv4  . . . . . . .   7
4.  Lack of Checksum Neutrality  . . . . . . . . . . . . . . . . .   8
5.  Security Considerations  . . . . . . . . . . . . . . . . . . .   8
6.  IANA Considerations  . . . . . . . . . . . . . . . . . . . . .   8
7.  Acknowledgements  . . . . . . . . . . . . . . . . . . . . . .   8
8.  References  . . . . . . . . . . . . . . . . . . . . . . . . .   8
  8.1.  Normative References  . . . . . . . . . . . . . . . . . .   8
  8.2.  Informative References  . . . . . . . . . . . . . . . . .   9
Appendix A.  Use Cases  . . . . . . . . . . . . . . . . . . . . .   9
  A.1.  464XLAT  . . . . . . . . . . . . . . . . . . . . . . . .   9
  A.2.  IVI  . . . . . . . . . . . . . . . . . . . . . . . . . .  10
  A.3.  SIIT-DC  . . . . . . . . . . . . . . . . . . . . . . . .  10
Appendix B.  Example IP Address Translations  . . . . . . . . . .  11
Author's Address  . . . . . . . . . . . . . . . . . . . . . . . .  12

1.  Introduction

The Stateless IP/ICMP Translation Algorithm (SIIT) [RFC6145] specifies that when translating IPv4 addresses to IPv6 and vice versa, all addresses must be translated using the algorithm specified in [RFC6052]. This document specifies an alternative to the [RFC6052] algorithm, where IP addresses are translated according to a table of Explicit Address Mappings configured on the stateless translator. This removes the previous constraint that IPv6 nodes that communicate with IPv4 nodes through SIIT must be configured with IPv4-translatable IPv6 addresses.
The Explicit Address Mapping Table does not replace [RFC6052]. For most use cases, it is expected that both algorithms are used in concert. The Explicit Address Mapping algorithm is used only when a mapping matching the address to be translated exists. If no matching mapping exists, the [RFC6052] algorithm will be used instead. Thus, when translating an individual IP packet, an SIIT implementation might translate one of the two IP address fields according to an EAM, while the other IP address field is translated according to [RFC6052].

1.1. Terminology

This document makes use of the following terms:

- **EAM**: An Explicit Address Mapping, as specified in Section 3.2.
- **EAMT**: The Explicit Address Mapping Table, as specified in Section 3.1.
- **SIIT**: The Stateless IP/ICMP Translation algorithm, as specified in [RFC6145].
- **IPv4-converted IPv6 addresses**: As defined in Section 1.3 of [RFC6052].
- **IPv4-translatable IPv6 addresses**: As defined in Section 1.3 of [RFC6052].

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].

2. Problem Statement

Section 3.2.1 of [RFC6144] notes that "stateless translation mechanisms typically put constraints on what IPv6 addresses can be assigned to IPv6 nodes that want to communicate with IPv4 destinations using an algorithmic mapping". In practice, this means that the IPv6 nodes must be configured with IPv4-translatable IPv6 addresses. For the reasons discussed below, some environments may find that the use of IPv4-translatable IPv6 addresses is not desired or even possible.

**Limited availability:**

The number of IPv4-translatable IPv6 addresses available to an operator is equal to the number of IPv4 addresses he assigns to
IPv4 addresses are scarce, and as a result an operator might not have enough IPv4-translatable IPv6 addresses to number his entire IPv6 infrastructure.

Restricted format:
IPv4-translatable IPv6 addresses must conform to the format specified in Section 2.2 of [RFC6052]. This format is not compatible with other common IPv6 address formats, such as the EUI-64 based IPv6 address format used by IPv6 Stateless Address Autoconfiguration [RFC4862].

An operator could overcome the above two problems by building an IPv6 network using regular (non-IPv4-translatable) IPv6 addresses, and assign IPv4-translatable IPv6 addresses as secondary addresses on the nodes that want to communicate with IPv4 nodes through SIIT only. However, doing so may result in a new set of undesired properties:

Routing complexity:
The IPv4-translatable IPv6 addresses must be routed throughout the IPv6 network separately from the primary (non-IPv4-translatable) IPv6 addresses used by the nodes. It might be impossible to aggregate these routes, as two adjacent IPv4-translatable IPv6 addresses might not be assigned to two adjacent IPv6 nodes. As a result, in order to support SIIT, the IPv6 network might need to carry a large number of extraneous routes. These routes must be separately injected into the IPv6 routing topology somehow. Any intermediate devices in the IPv6 network such as a firewall might require special configuration in order to treat the IPv4-translatable IPv6 address the same as the primary IPv6 address, for example by requiring that any ACL entries involving the primary IPv6 address of a node must be duplicated.

Operational complexity:
The IPv4-translatable IPv6 addresses must not only be assigned to the IPv6 nodes participating in SIIT; all applications and services on those nodes must also be configured to use them. For example, if the IPv6 node is a load balancer, it might require a separate Virtual Server definition using the IPv4-translatable IPv6 address in addition to one using the service’s primary IPv6 address. A web server might require specific configuration to listen for connections on both the IPv4-translatable and the primary IPv6 address. A High-Availability cluster service must be set up to fail over both addresses between cluster nodes, and depending on how the IPv6 network learns the location of the IPv4-translatable IPv6 address, the fail-over mechanism used for the two addresses might be completely different. Service monitoring must be done for both the IPv4-translatable and the primary IPv6 address, and any trouble-shooting procedures must be extended to involve both addresses.

In short, the use of IPv4-translatable IPv6 addresses in parallel with regular IPv6 addresses is in many ways analogous to the use of Dual Stack [RFC4213]. While no actual IPv4 packets are used, the IPv4-translatable IPv6 addresses creates a secondary "stack" in the infrastructure that must be treated and operated separately from the primary one. This increases the complexity of the overall infrastructure, in turn increasing operational overhead, and reducing reliability. An operator who for such reasons finds the use of Dual Stack unappealing, might feel the same way about using SIIT with IPv4-translatable IPv6 addresses.

3. Explicit Address Mapping Algorithm

This normative section defines the EAM algorithm. SIIT implementations are REQUIRED to support the specifications herein.

3.1. Explicit Address Mapping Table

An SIIT implementation MUST include an Explicit Address Mapping Table (EAMT). By default, the EAMT SHOULD be empty. The operator MUST be able to populate the EAMT using the implementation’s normal configuration interfaces. The implementation MAY additionally support other ways of populating the EAMT.

The EAMT consists of the following columns:

IPv4 Prefix
IPv6 Prefix
SIIT implementations MAY include other columns in order to support proprietary extensions to the EAM algorithm.

Throughout this document, figures representing the EAMT contain an Index column using the pound sign as the header. This column is not a required part of this specification; it is included only as a convenience to the reader.

### 3.2. Explicit Address Mapping Specification

An EAM consists of an IPv4 Prefix and an IPv6 Prefix. The prefix length MAY be omitted, in which case the implementation MUST assume it to be 32 for IPv4 and 128 for IPv6. Figure 1 illustrates an EAMT containing examples of valid EAMs.

**Example EAMT**

<table>
<thead>
<tr>
<th>#</th>
<th>IPv4 Prefix</th>
<th>IPv6 Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>192.0.2.1</td>
<td>2001:db8:aaaa::</td>
</tr>
<tr>
<td>2</td>
<td>192.0.2.2/32</td>
<td>2001:db8:bbbb::b/128</td>
</tr>
<tr>
<td>3</td>
<td>192.0.2.16/28</td>
<td>2001:db8:cccc::/124</td>
</tr>
<tr>
<td>4</td>
<td>192.0.2.128/26</td>
<td>2001:db8:dddd::/64</td>
</tr>
<tr>
<td>5</td>
<td>192.0.2.192/31</td>
<td>64:ff9b::/127</td>
</tr>
</tbody>
</table>

**Figure 1**

An EAM’s IPv4 Prefix value MUST have an identical or smaller number of suffix bits than its corresponding IPv6 Prefix value.

Overlapping EAMs SHOULD be considered an error, and attempts to insert them into the EAMT SHOULD be blocked. The behaviour of an SIIT implementation when overlapping EAMs are present in the EAMT is left undefined.

When translating a packet between IPv4 and IPv6, an SIIT implementation MUST individually translate each IP address it encounters in the packet’s IP headers (including any IP headers contained within ICMP errors) according to Section 3.3.

### 3.3. IP Address Translation Procedure

This section describes step-by-step how an SIIT implementation translates addresses between IPv4 and IPv6. Only the outcome of the algorithm described should be considered normative, that is, an SIIT implementation MAY implement the exact procedure differently than
what is described here, but the outcome of the algorithm MUST be the same.

For concrete examples of IP addresses translations, refer to Appendix B.

3.3.1. Address Translation Steps: IPv4 to IPv6

1. The EAMT is searched for an EAM entry containing an IPv4 Prefix identical to that of the IPv4 address being translated. The IPv4 Prefix and IPv6 Prefix values of the EAM entry found is from now on referred to as EAM4 and EAM6, respectively.

2. If no matching EAM entry is found, the EAM algorithm is aborted. The SIIT implementation MUST proceed to translate the address in accordance with [RFC6145] (and its updates).

3. The prefix bits of EAM4 are removed from IPv4 address being translated. The remaining suffix bits from the IPv4 address being translated are stored in a temporary buffer.

4. The prefix bits of EAM6 are prepended to the temporary buffer.

5. If the temporary buffer at this point does not contain a 128-bit value, it is padded with trailing zeroes so that it reaches a length of 128 bits.

6. The contents of the temporary buffer is the translated IPv6 address.

3.3.2. Address Translation Steps: IPv6 to IPv4

1. The EAMT is searched for an EAM entry containing an IPv6 Prefix identical to that of the IPv6 address being translated. The IPv4 Prefix and IPv6 Prefix values of the EAM entry found is from now on referred to as EAM4 and EAM6, respectively.

2. If no matching EAM entry is found, the EAM algorithm is aborted. The SIIT implementation MUST proceed to translate the address in accordance with [RFC6145] (and its updates).

3. The prefix bits of EAM6 are removed from IPv6 address being translated. The remaining suffix bits from the IPv6 address being translated are stored in a temporary buffer.

4. The prefix bits of EAM4 are prepended to the temporary buffer.
5. If the temporary buffer at this point does not contain a 32-bit value, any trailing bits are discarded so that the buffer is reduced to a length of 32 bits.

6. The contents of the temporary buffer is the translated IPv4 address.

4. Lack of Checksum Neutrality

When one or both of the address fields in an IP/ICMP packet are translated according to EAM, the translation can not be relied upon to be checksum neutral, even if the well-known prefix 64:ff9b::/96 is used. This consideration is discussed in more detail in Section 4.1 of [RFC6052].

5. Security Considerations

The EAM algorithm does not introduce any new security issues beyond those that are already discussed in Section 7 of [RFC6145].

6. IANA Considerations

This draft makes no request of the IANA. The RFC Editor may remove this section prior to publication.

7. Acknowledgements

This document was conceived due to comments made by Dave Thaler in the v6ops session at IETF 91 as well as e-mail discussions between Fred Baker and the author.

Valuable reviews, suggestions, and other feedback was given by Cameron Byrne, Brian E Carpenter, Alberto Leiva, and Andrew Yourtchenko.

8. References

8.1. Normative References


8.2. Informative References


Appendix A. Use Cases

The following subsections lists some use cases that at the time of writing leverage SIIT with the EAM algorithm.

A.1. 464XLAT

When the CLAT component in the 464XLAT [RFC6877] architecture does not have a dedicated IPv6 prefix assigned, it may instead use "one interface IPv6 address that is claimed by the CLAT". This IPv6 address might not be IPv4-translatable. If this is the case, the CLAT essentially implements the EAM algorithm using an EAMT as follows (assuming the CLAT’s IPv4 address is picked from the IPv4 Service Continuity Prefix [RFC7335]):

Example EAMT for an 464XLAT CLAT
In this particular use case, the EAM algorithm is used to translate IPv6 destination addresses to IPv4, and conversely, IPv4 source addresses to IPv6. Other addresses are translated using [RFC6052]. Note that this is the exact opposite of the SIIT-DC use case (Appendix A.3).

A.2. IVI

IVI [RFC6219] describes a stateless translation model that embeds IPv4 addresses in a 40-bit translation prefix where bits 33-40 are required to be 1. The embedded IPv4 address is located in bits 41-72 of the IPv6 address. Bits 73-128 are required to be 0.

The location of the eight least significant IPv4 address bits makes the IVI address mapping differ from [RFC6052].

Example EAMT for IVI

<table>
<thead>
<tr>
<th>#</th>
<th>IPv4 Prefix</th>
<th>IPv6 Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0.0.0/0</td>
<td>2001:db8:ff00::/40</td>
</tr>
</tbody>
</table>

In this particular use case, all addresses are translated according to the EAM algorithm. In other words, [RFC6052] mapping is not used at all.

A.3. SIIT-DC
SIIT-DC [I-D.anderson-v6ops-siit-dc] describes the use of SIIT to facilitate connectivity from the IPv4 Internet to services hosted in an IPv6-only data centre. In order to avoid the constraints relating to the use of IPv4-translatable IPv6 addresses discussed in Section 2 the stateless IPv4/IPv6 translators are provisioned with an EAMT containing one entry per IPv6-only service that are to be made available from the IPv4 Internet, for example (assuming 2001:db8:aaaa::1 and 2001:db8:bbbb::1 are assigned to load balancers or servers that provides the IPv6-only services in question):

```
+---+--------------+----------------------+
| # | IPv4 Prefix  |     IPv6 Prefix      |
|---+--------------+----------------------|
| 1 | 192.0.2.1/32 | 2001:db8:aaaa::1/128 |
| 2 | 192.0.2.2/32 | 2001:db8:bbbb::1/128 |
```

Figure 4

In this particular use case, the EAM algorithm is used to translate IPv4 destination addresses to IPv6, and conversely, IPv6 source addresses to IPv4. Other addresses are translated using [RFC6052]. Note that this is the exact opposite of the 464XLAT use case (Appendix A.1).

Appendix B. Example IP Address Translations

Figure 5 demonstrates how a set of example IP addresses are translated given the example EAMT in Figure 1. Implementors may use the examples given to develop test cases to validate correct operation. Note that the address translations are bidirectional, so a single row in the table describes two address translations: IPv4 to IPv6, and IPv6 to IPv4.

It is also assumed that the [RFC6052] translation prefix is configured to be 64:ff9b::/96.

```
+----------------------------------+
| # | IPv4 Prefix  |     IPv6 Prefix      |
|----+--------------+----------------------|
| 1  | 192.0.2.1/32 | 2001:db8:aaaa::1/128 |
| 2  | 192.0.2.2/32 | 2001:db8:bbbb::1/128 |
```
<table>
<thead>
<tr>
<th>IPv4 Address</th>
<th>IPv6 Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.0.2.1</td>
<td>2001:db8:aaaa::</td>
<td>According to EAM #1</td>
</tr>
<tr>
<td>192.0.2.2</td>
<td>2001:db8:bbbb::b</td>
<td>According to EAM #2</td>
</tr>
<tr>
<td>192.0.2.16</td>
<td>2001:db8:cccc::</td>
<td>According to EAM #3</td>
</tr>
<tr>
<td>192.0.2.24</td>
<td>2001:db8:cccc::8</td>
<td>According to EAM #3</td>
</tr>
<tr>
<td>192.0.2.31</td>
<td>2001:db8:cccc::f</td>
<td>According to EAM #3</td>
</tr>
<tr>
<td>192.0.2.128</td>
<td>2001:db8:dddd::</td>
<td>According to EAM #4</td>
</tr>
<tr>
<td>192.0.2.152</td>
<td>2001:db8:dddd:0:6000::</td>
<td>According to EAM #4</td>
</tr>
<tr>
<td>192.0.2.183</td>
<td>2001:db8:dddd:0:dc00::</td>
<td>According to EAM #4</td>
</tr>
<tr>
<td>192.0.2.191</td>
<td>2001:db8:dddd:0:fc00::</td>
<td>According to EAM #4</td>
</tr>
<tr>
<td>192.0.2.193</td>
<td>64:ff9b::1</td>
<td>According to EAM #5</td>
</tr>
<tr>
<td>192.0.2.200</td>
<td>64:ff9b::c000:2c8</td>
<td>According to RFC 6052</td>
</tr>
</tbody>
</table>

Figure 5

Author’s Address

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
Observations on IPv6 EH Filtering in the Real World
draft-gont-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 filtered in the Internet (as measured in August 2014), and where in the network such filtering 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.

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 9, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.
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 IPv6 Extension Headers are filtered in the Internet, as measured in August 2014 (pending operational advice in this area).
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 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 such list)
- Name servers (NS -> AAAA of such list)

IPv6 addresses other than global unicast addresses and duplicate 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 performed:

- 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 Destination Options Header and 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, bugs on 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, our measurements result in a percentage *range* (see Appendix B.2). In the following tables, the values shown within parentheses represent the estimated range of possibility that when a packet is dropped, the packet drop occurs in an AS other than the destination AS.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DO8</th>
<th>HBH8</th>
<th>FH512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Webservers</td>
<td>11.88%</td>
<td>40.70%</td>
<td>30.51%</td>
</tr>
<tr>
<td></td>
<td>(17.60%-20.80%)</td>
<td>(31.43%-40.00%)</td>
<td>(5.08%-6.78%)</td>
</tr>
<tr>
<td>Mailservers</td>
<td>17.07%</td>
<td>48.86%</td>
<td>39.17%</td>
</tr>
<tr>
<td></td>
<td>(6.35%-26.98%)</td>
<td>(40.50%-65.42%)</td>
<td>(2.91%-12.73%)</td>
</tr>
<tr>
<td>Nameservers</td>
<td>15.37%</td>
<td>43.25%</td>
<td>38.55%</td>
</tr>
<tr>
<td></td>
<td>(14.29%-33.46%)</td>
<td>(42.49%-72.07%)</td>
<td>(3.90%-13.96%)</td>
</tr>
</tbody>
</table>

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, between 17.60%-20.80% of them get dropped at an AS other than the destination AS".
<table>
<thead>
<tr>
<th>EH Type</th>
<th>Webservers</th>
<th>Mailservers</th>
<th>Nameservers</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO8</td>
<td>11.88%</td>
<td>17.07%</td>
<td>15.37%</td>
</tr>
<tr>
<td></td>
<td>(17.60%-20.80%)</td>
<td>(6.35%-26.98%)</td>
<td>(14.29%-33.46%)</td>
</tr>
<tr>
<td>HBH8</td>
<td>40.70%</td>
<td>48.86%</td>
<td>43.25%</td>
</tr>
<tr>
<td></td>
<td>(31.43%-40.00%)</td>
<td>(40.50%-65.42%)</td>
<td>(42.49%-72.07%)</td>
</tr>
<tr>
<td>FH512</td>
<td>30.51%</td>
<td>39.17%</td>
<td>38.55%</td>
</tr>
<tr>
<td></td>
<td>(5.08%-6.78%)</td>
<td>(2.91%-12.73%)</td>
<td>(3.90%-13.96%)</td>
</tr>
</tbody>
</table>

Table 2: WIPv6LD dataset: Packet drop rate for different EH types, and estimated percentage of dropped packets that were deemed to be dropped in a different AS (lower, in parentheses)

NOTE: This table contains the same information as Table 1, but makes it easier to obtain the drop rates for each EH type. Each cell should be read in exactly the same way as each cell in Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DO8</th>
<th>HBH8</th>
<th>FH512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Webservers</td>
<td>10.91%</td>
<td>39.03%</td>
<td>28.26%</td>
</tr>
<tr>
<td></td>
<td>(46.52%-53.23%)</td>
<td>(36.90%-46.35%)</td>
<td>(53.64%-61.43%)</td>
</tr>
<tr>
<td>Mailservers</td>
<td>11.54%</td>
<td>45.45%</td>
<td>35.68%</td>
</tr>
<tr>
<td></td>
<td>(2.41%-21.08%)</td>
<td>(41.27%-61.13%)</td>
<td>(3.15%-10.92%)</td>
</tr>
<tr>
<td>Nameservers</td>
<td>21.33%</td>
<td>54.12%</td>
<td>55.23%</td>
</tr>
<tr>
<td></td>
<td>(10.27%-56.80%)</td>
<td>(50.64%-81.00%)</td>
<td>(5.66%-32.23%)</td>
</tr>
</tbody>
</table>

Table 3: 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)
Table 4: Alexa’s top 1M sites dataset: Packet drop rate for different EH types, and estimated percentage of dropped packets that were deemed to be dropped in a different AS (lower, in parentheses)

<table>
<thead>
<tr>
<th>EH Type</th>
<th>Webservers</th>
<th>Mailservers</th>
<th>Nameservers</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO8</td>
<td>10.91%</td>
<td>11.54%</td>
<td>21.33%</td>
</tr>
<tr>
<td></td>
<td>(46.52%-53.23%)</td>
<td>(2.41%-21.08%)</td>
<td>(10.27%-56.80%)</td>
</tr>
<tr>
<td>HBH8</td>
<td>39.03%</td>
<td>45.45%</td>
<td>54.12%</td>
</tr>
<tr>
<td></td>
<td>(36.90%-46.35%)</td>
<td>(41.27%-61.13%)</td>
<td>(50.64%-81.00%)</td>
</tr>
<tr>
<td>FH512</td>
<td>28.26%</td>
<td>35.68%</td>
<td>55.23%</td>
</tr>
<tr>
<td></td>
<td>(53.64%-61.43%)</td>
<td>(3.15%-10.92%)</td>
<td>(5.66%-32.23%)</td>
</tr>
</tbody>
</table>

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 occur 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 filtered 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) Mark Andrews, Fred Baker, Brian Carpenter and Tatuya Jinmei 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 and Go6 Lab <http://go6lab.si/> 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

6.2. Informative References

[Gont-Chown-IEPG89]

[Gont-IEPG88]

[IANA-PORT-NUMBERS]

[IPv6-Toolkit]

[Linkova-Gont-IEPG90]

[PMTUD-Blackholes]

[RFC5927]

[RFC6980]

[RFC7045]

[RFC7113]

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 an 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 domains can be obtained with:

$ cat top-1m.txt | script6 get-aaaa > top-1m-web-aaaa.txt
The AAAA records corresponding to the mail servers 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 web servers:

# 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 web servers:

# 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 web servers:

# 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 mail servers:

```
# 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:
In order to compute the statistics corresponding to our measurements of FH512 with the list of webservers:

$ cat top-1m-web-aaaa-hbh8-m.txt | script6 get-trace6-stats > top-1m-web-aaaa-hbh8-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 both 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 filtering 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 "percentage ranges" (rather than a specific ratio): 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 prove 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] 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
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.

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 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
1. Introduction

This document discusses certain choices that arise when designing a IPv6-only or dual-stack network. The focus is on routing-related design choices that do not usually 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.

This document does not present advice on strategies for adding IPv6 to a network, nor does it discuss transition mechanisms. For advice 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 details of ULA usage is also not discussed; for this the reader is referred to [I-D.ietf-v6ops-ula-usage-recommendations].

Finally, this document focuses on unicast routing design only and does not cover multicast or the issues involved in running MPLS over IPv6 transport.

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. Interfaces

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

o 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.

2.1.2. Interfaces with Only Link-Local Addresses?

As noted in the introduction, this document does not cover the ins
and outs around creating an IPv6 addressing plan. However, there is
one fundamental question in this area that often arises: Should an
interface:

a. Use only a link-local address ("unnumbered"), OR

b. Have global and/or unique-local addresses assigned in addition to
   the link-local?

There are two advantages in interfaces with only link-local addresses
("unnumbered interfaces"). The first advantage is ease of
configuration. In a network with a large number of unnumbered
interfaces, the operator can just enable an IGP on each router,
without going through the tedious process of assigning and tracking
the addresses for each interface. The second advantage is security.
Since packets with Link-Local destination addresses 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 this advantage are various disadvantages to interfaces
with only link-local addresses:

o It is not possible to ping an interface that has only a link-local
  address 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 unnumbered interface will return the loopback or system 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].

- 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].

- 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 numbered interfaces (option b).
2.2. Static Routes

2.2.1. 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.

2.3. IGPs
2.3.1. IGP Choice

One of the main decisions for an IPv6 implementer is the choice of IGP (Interior Gateway Protocol) within the network. The primary options are OSPF [RFC2328] [RFC5340] or IS-IS [RFC5120] [RFC5308], though some operators may consider RIP [RFC2080] or non-standardized protocols. Here we limit our discussion to the pros and cons of OSPF vs. IS-IS.

The discussion in this section revolves around the options in the table below:

<table>
<thead>
<tr>
<th>Option</th>
<th>IGP for IPv4</th>
<th>IGP for IPv6</th>
<th>Known to work well</th>
<th>Hard separation</th>
<th>Similar configuration possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>IS-IS</td>
<td>IS-IS</td>
<td>YES</td>
<td>-</td>
<td>YES</td>
</tr>
<tr>
<td>b</td>
<td>IS-IS</td>
<td>OSPFv3</td>
<td>-</td>
<td>YES</td>
<td>-</td>
</tr>
<tr>
<td>c</td>
<td>OSPFv2</td>
<td>IS-IS</td>
<td>YES</td>
<td>YES</td>
<td>-</td>
</tr>
<tr>
<td>d</td>
<td>OSPFv2</td>
<td>OSPFv3</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>e</td>
<td>OSPFv3</td>
<td>IS-IS</td>
<td>-</td>
<td>YES</td>
<td>-</td>
</tr>
<tr>
<td>f</td>
<td>OSPFv3</td>
<td>OSPFv3</td>
<td>-</td>
<td>-</td>
<td>YES</td>
</tr>
</tbody>
</table>

Three of the options above are marked as "Known to work well". These options have seen significant deployments and are generally considered to be good choices. The other options represent valid choices, but have not seen widespread use, so it is hard to offer comments on how well they work. In particular, options (e) and (f) use OSPFv3 to route IPv4 [RFC5838], which is still rather new and untested.

A number of options are marked "Hard separation". These options use a different IGP for IPv4 vs IPv6. With these options, a problem with routing IPv6 is unlikely to affect IPv4 or visa-versa.
Three options 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 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.

With option (a), there is an additional choice of whether to run IS-IS in single-topology mode (where IPv4 and IPv6 share a single topology and a single set of link costs[RFC5308]) or multi-topology mode (where IPv4 and IPv6 have separate topologies and potentially different link costs[RFC5120]). A big problem with single-topology mode is that it cannot easily accommodate devices that support IPv4-only or IPv6-only. Thus, today there is general agreement that multi-topology is the right choice as this gives the greatest flexibility in network design.

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. BGP

2.4.1. Which Transport for Which Routes?

BGP these days is multi-protocol. It can carry routes from many different families, and it can do this 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 routes should be carried over sessions using IPv4 transport and which should be carried over sessions using IPv6 transport.

To answer this question, consider the following table:
<table>
<thead>
<tr>
<th>Route Family</th>
<th>Transport</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlabeled IPv4</td>
<td>IPv4</td>
<td>Works well</td>
</tr>
<tr>
<td>Unlabeled IPv4</td>
<td>IPv6</td>
<td>Next-hop issues</td>
</tr>
<tr>
<td>Unlabeled IPv6</td>
<td>IPv4</td>
<td>Next-hop issues</td>
</tr>
<tr>
<td>Unlabeled IPv6</td>
<td>IPv6</td>
<td>Works well</td>
</tr>
<tr>
<td>Labeled IPv4</td>
<td>IPv4</td>
<td>Works well</td>
</tr>
<tr>
<td>Labeled IPv4</td>
<td>IPv6</td>
<td>Next-hop issues</td>
</tr>
<tr>
<td>Labeled IPv6</td>
<td>IPv4</td>
<td>(6PE) Works well</td>
</tr>
<tr>
<td>Labeled IPv6</td>
<td>IPv6</td>
<td>Needs MPLS over IPv6</td>
</tr>
<tr>
<td>VPN IPv4</td>
<td>IPv4</td>
<td>Works well</td>
</tr>
<tr>
<td>VPN IPv4</td>
<td>IPv6</td>
<td>Next-hop issues</td>
</tr>
<tr>
<td>VPN IPv6</td>
<td>IPv4</td>
<td>(6VPE) Works well</td>
</tr>
<tr>
<td>VPN IPv6</td>
<td>IPv6</td>
<td>Needs MPLS over IPv6</td>
</tr>
</tbody>
</table>

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 widely supported and are generally recommended.
For combinations marked "Next-hop issues", these combinations are less-widely supported and when supported, often have next-hop issues. That is, the next-hop address is typically a v4-mapped IPv6 address, which is based on some IPv4 address on the sending router. This v4-mapped IPv6 address is often not reachable by default using IPv6 routing. One common solution to this problem is to use routing policy to change the next-hop to a different IPv6 address.

For combinations marked as "Needs MPLS over IPv6", these require MPLS over IPv6 for full support, though special policy configuration may allow them to be used with MPLS over IPv4.

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 scenarios in more detail.

2.4.1.1. 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, operators today most commonly use two BGP sessions: one session is transported over IPv4 and carries the unlabeled IPv4 routes, while the second session is transported over IPv6 and carries the unlabeled IPv6 routes.

There are several reasons for this choice:

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

- This avoids the next-hop problem described in note 1 above.

- 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.

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

2.4.1.2. BGP sessions for Labeled or VPN Routes

In these scenarios, it is most common today to carry both the IPv4 and IPv6 routes over sessions transported over IPv4. This can be done with either: (a) one session carrying both route families, or (b) two sessions, one for each family.

Using a single session is usually appropriate 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.

2.4.2. 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:
o Using link-local addresses only works for single-hop eBGP sessions; it does not work for multi-hop sessions.

o 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).

o Operators and their tools are used to referring to eBGP sessions by address only, something that is not possible with link-local addresses.

o 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.

o On some products, an eBGP session using a link-local address is more complex to configure than a session that uses a global address.

o 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).

o 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 through 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]

[I-D.ietf-idr-dynamic-cap]

[I-D.ietf-v6ops-ula-usage-recommendations]

[ISO10589]


Authors’ Addresses

Philip Matthews
Alcatel-Lucent
600 March Road
Ottawa, Ontario K2K 2E6
Canada

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

Victor Kuarsingh
Dyn
150 Dow Street
Manchester, NH 03101
USA

Email: victor@jvknet.com
Abstract

This document calls attention to the problem of delivering ICMPv6 type 2 "Packet Too Big" (PTB) messages to the intended destination in ECMP load balanced, or anycast network architectures. It discusses operational mitigations that can be employed to address this class of failure.

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 2, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction .............................................. 2
2. Problem ..................................................... 2
3. Mitigation .................................................. 4
   3.1. Alternatives ......................................... 5
   3.2. Implementation ....................................... 5
4. Improvements .............................................. 6
5. Acknowledgements .......................................... 7
6. IANA Considerations ....................................... 7
7. Security Considerations ..................................... 7
8. Informative References ..................................... 7

1. Introduction

Operators of popular Internet services face complex challenges
associated with scaling their infrastructure. One approach is to
utilize equal-cost multi-path (ECMP) routing to perform stateless
distribution of incoming TCP or UDP sessions to multiple servers or
to middle boxes such as load balancers. Distribution of traffic in
this manner presents a problem when dealing with ICMP signaling.
Specifically, an ICMP error is not guaranteed to hash via ECMP to the
same destination as its corresponding TCP or UDP session. A case
where this is particularly problematic operationally is path MTU
discovery (PMTUD).

2. Problem

A common application for stateless load balancing of TCP or UDP flows
is to perform an initial subdivision of flows in front of a stateful
load balancer tier or multiple servers, so that the workload becomes
divided into manageable fractions of the total number of flows. The
flow division is performed using ECMP forwarding and a stateless but
sticky algorithm for hashing across the available paths. This
nexthop selection for the purposes of flow distribution is a
constrained form of anycast topology, where all anycast destinations
are equidistant from the upstream router responsible for making the
last next-hop forwarding decision before the flow arrives on the
destination device. In this approach, the hash is performed across
some set of available protocol headers. Typically, these headers may
include all or a subset of (IPv6)Flow-Label, IP-source, IP-
destination, protocol, source-port, destination-port and potentially
others such as ingress interface.
A problem common to this approach of distribution through hashing is impact on path MTU discovery. An ICMPv6 type 2 PTB message generated on an intermediate device for a packet sent from an a server that is part of an ECMP load balanced service to a client, will have the load-balanced anycast address as the destination and would be statelessly load balanced to one of the servers. While the ICMPv6 PTB message contains as much of the packet that could not be forwarded as possible, the payload headers are not considered into the forwarding decision and are ignored. Because the PTB message is not identifiable as part of the original flow by the IP or upper layer packet headers the results of the ICMPv6 ECMP hash are unlikely to be hashed to the same nexthop as packets matching TCP or UDP ECMP hash.

An example packet flow and topology follow.

```
ptb --> router ecmp --> nexthop L4/L7 load balancer --> destination
  \     \                      \
  \     \                      \
  \     \                      \\--> load balancer 1 --->
  \     \                      \--> load balancer 2 ---> load-balanced service
  \     \                      \--> load balancer N --->
```

Figure 1

The router ECMP decision is used because it is part of the forwarding architecture, can be performed at line rate, and does not depend on shared state or coordination across a distributed forwarding system which may include multiple linecards or routers. The ECMP routing decision is deterministic with respect to packets having the same computed hash.

A typical case where ICMPv6 PTB messages are received at the load balancer is a case where the path MTU from the client to the load balancer is limited by a tunnel in which the client itself is not aware of. In the common case of a TCP flow where TLS is employed, the first packet sent from the server that is likely to exceed a tunnel MTU lower than that specified by the MSS on the client and the load balancer/server is the TLS ServerHello and certificate.

Direct experience says that the frequency of PTB messages is small compared to total flows. One possible conclusion being that tunneled IPv6 deployments that cannot carry 1500 mtu packets are relatively rare. Techniques employed by clients such as happy-eyeballs may actually contribute some amelioration to the IPv6 client experience by preferring IPv4 in cases that might be identified as failures.
Still, the expectation of operators is that PMTUD should work and that unnecessary breakage of client traffic should be avoided.

A final observation regarding server tuning is that it is not always possible even if it is potentially desirable to be able to independently set the TCP MSS for different address families on end-systems.

The problem as described does also impact IPv4; however, the ability to fragment on wire at tunnel ingress points and the relative rarity of sub-1500 byte MTUs that are not coupled to changes in client behavior (for example, endpoint VPN clients set the tunnel interface MTU accordingly for performance reasons) makes the problem sufficiently rare that some existing deployments simply choose to ignore it.

3. Mitigation

Mitigation of the potential for PTB messages to be mis-delivered involves ensuring that an ICMPv6 error message is distributed to the same anycast server responsible for the flow for which the error is generated. Ideally Mitigation could be done by the mechanism hosts use to identify the flow, by looking into the payload of the ICMPv6 message (to determine which TCP flow it was associated with) before making a forwarding decision. Because the encapsulated IP header occurs at a fixed offset in the icmp message it is not outside the realm of possibility that routers with sufficient header processing capability could parse that far into the payload. Employing a mediation device that handles the parsing and distribution of PTB messages after policy routing or on each load-balancer/server is a possibility.

Another mitigation approach is predicated upon distributing the PTB message to all anycast servers under the assumption that the one for which the message was intended will be able to match it to the flow and update the route cache with the new MTU, devices not able to match the flow will discard these packets. Such distribution has potentially significant implications for resource consumption and the potential for self-inflicted denial-of-service if not carefully employed. Fortunately, in real-world-deployment we have observed that, the number of flows for which this problem occurs is relatively small (example, 10 or fewer pps on 1Gb/s or more worth of https traffic) and sensible ingress rate limiters which will discard excessive message volume can be applied to protect even very large anycast server tiers with the potential for fallout only under circumstances of deliberate duress.
3.1. Alternatives

As an alternative, it may be appropriate to lower the TCP MSS to 1220 in order to accommodate 1280 byte MTU. We consider this undesirable as hosts may not be able to independently set TCP MSS by address-family thereby impacting IPv4, or alternatively that it relies on a middle-box to clamp the MSS independently from the end-systems.

3.2. Implementation

1. Filter-based-forwarding matches next-header ICMPv6 type-2 and matches a next-hop on a particular subnet directly attached to both border routers. The filter is policed to reasonable limits (we chose 1000pps).

2. Filter is applied on input side of all external interfaces

3. A proxy located at the next-hop forwards ICMPv6 type-2 packets received at the next-hop to an Ethernet broadcast address (example ff:ff:ff:ff:ff:ff) on all specified subnets. This was necessitated by router inability (in IPv6) to forward the same packet to multiple unicast next-hops.

4. Anycast servers receive the PTB error and process packet as needed.

A simple Python scapy script that can perform the ICMPv6 proxy reflection is included.
#!/usr/bin/python
from scapy.all import *

IFACE_OUT = ["p2p1", "p2p2"]

def icmp6_callback(pkt):
    if pkt.haslayer(IPv6) and (ICMPv6PacketTooBig in pkt) \
    and pkt[Ether].dst != 'ff:ff:ff:ff:ff:ff':
        del(pkt[Ether].src)
        pkt[Ether].dst = 'ff:ff:ff:ff:ff:ff'
        pkt.show()
        for iface in IFACE_OUT:
            sendp(pkt, iface=iface)

def main():
    sniff(prn=icmp6_callback, filter="icmp6 \n    and (ip6[40+0] == 2)", store=0)

if __name__ == '__main__':
    main()

This example script listens on all interfaces for IPv6 PTB errors
being forwarded using filter-based-forwarding. It removes the
existing Ethernet source and rewrites a new Ethernet destination of
the Ethernet broadcast address. It then sends the resulting frame
out the p2p1 and p2p2 interfaces where our anycast servers reside.

Alternatively, network designs in which a common layer2 network
exists could rewrite the destination on the end system, for example
in using iptables before forwarding the packet back to the network
containing all of the server or load balancer interfaces.

4. Improvements

There are several ways that improvements could be made to the
situation with respect to ECMP load balancing of ICMPv6 PTB.

1. Routers with sufficient capacity within the lookup process could
   parse all the way through the L3 or L4 header in the ICMPv6
   payload beginning at bit offset 32 of the ICMP header. By
   reordering the elements of the hash to match the inward direction
   of the flow, the PTB error could be directed to the same next-hop
   as the incoming packets in the flow.

2. The FIB could be programmed with a multicast distribution tree
   that included all of the necessary next-hops.

3. Ubiquitous implementation of RFC 4821 [RFC4821] Packetization Layer Path MTU Discovery would probably go a long way towards reducing dependence on ICMPv6 PTB.

5. Acknowledgements

The authors would like to thank Mark Andrews, Brian Carpenter, Nick Hilliard and Ray Hunter, for review.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

The employed mitigation has the potential to greatly amplify the impact of a deliberately malicious sending of ICMPv6 PTB messages. Sensible ingress rate limiting can reduce the potential for impact; however, legitimate traffic may be lost once the rate limit is reached.

The proxy replication results in devices not associated with the flow that generated the PTB being recipients of an ICMPv6 message which contains a fragment of a packet. This could arguably result in information disclosure. Recipient machines should be in a common administrative domain.

8. Informative References


Authors’ Addresses

Matt Byerly
Fastly
Kapolei, HI
US

Email: mbyerly@zynga.com

Matt Hite
Evernote
Redwood City, CA
US

Email: mhite@hotmail.com
Joel Jaeggli
Fastly
Mountain View, CA
US

Email: joelja@gmail.com
SIIT-DC: Stateless IP/ICMP Translation for IPv6 Data Centre Environments
draft-ietf-v6ops-siit-dc-00

Abstract

This document describes SIIT-DC, an extension to the Stateless IP/ICMP Translation (SIIT) algorithm, that makes it ideally suited for use in IPv6 data centre environments. SIIT-DC simultaneously facilitates IPv6 deployment and IPv4 address conservation. The overall SIIT-DC architecture is described, as well as guidelines for operators. Finally, the normative implementation requirements are described, as a list of additions and changes to SIIT.

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 June 21, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ................................................. 3
   1.1. Motivation and Goals ................................. 3
       1.1.1. Single Stack IPv6 Operation ..................... 4
       1.1.2. Stateless Operation .............................. 4
       1.1.3. IPv4 Address Conservation ....................... 5
       1.1.4. No Loss of End User's IPv4 Source Address ...... 5
       1.1.5. Compatible with Standard IPv6 Implementations ... 5
       1.1.6. No Architectural Dependency on IPv4 ............. 6
2. Terminology ................................................. 6
3. Architectural Overview ..................................... 7
   3.1. DNS Configuration ..................................... 9
   3.2. Packet Flow ............................................. 9
4. Deployment Guidelines ...................................... 11
   4.1. Application Support for NAT .......................... 12
   4.2. Application Support for IPv6 .......................... 12
   4.3. Application Communication Pattern .................... 12
   4.4. Choice of Translation Prefix .......................... 13
   4.5. Routing Considerations ............................... 13
   4.6. Location of the SIIT-DC Gateways ..................... 14
   4.7. Migration from Dual Stack ............................. 15
   4.8. Packet Size and Fragmentation Considerations ......... 15
       4.8.1. IPv4/IPv6 Header Size Difference ................ 16
       4.8.2. IPv6 Atomic Fragments ............................ 16
       4.8.3. Minimum Path MTU Difference Between IPv4 and IPv6 . 17
5. Implementation Requirements ................................. 18
   5.1. Compliance with RFC6145 and RFC6052 .................. 18
   5.2. Static Address Mapping Function ....................... 19
   5.3. Support for Increasing the IPv6 Path MTU ............. 19
   5.4. Loop Prevention Mechanism ............................. 20
6. Acknowledgements ........................................... 20
7. Requirements Language ..................................... 20
8. IANA Considerations ....................................... 20
9. Security Considerations .................................... 20
   9.1. Mistaking the Translation Prefix for a Trusted Network . 21
   9.2. Packets Looping Through the SIIT-DC Function ....... 21
10. References ................................................. 21
    10.1. Normative References ............................... 21
    10.2. Informative References ............................. 21
Appendix A. Complete SIIT-DC topology example .................. 23
Appendix B. Comparison to Other Deployment Approaches ........ 26
   B.1. IPv4-only .............................................. 26
   B.2. IPv4-only + NAPT44 .................................... 26
   B.3. IPv4-only + NAT64 ..................................... 28
1. Introduction

SIIT-DC is an extension of SIIT [RFC6145] that provides a network-centric stateless translation service that allows a data centre operator or Internet Content Provider (ICP) to run a data centre network, servers, and applications using exclusively IPv6, while at the same time ensuring that end users that have only IPv4 connectivity will be able to continue to access the services and applications.

1.1. Motivation and Goals

Historically, dual stack [RFC4213] [RFC6883] has been the recommended way to transition from an IPv4-only environment to one capable of serving IPv6 users. However, for data centre operators and Internet content providers, dual stack operation has a number of disadvantages compared to single stack operation. In particular, running two protocols rather than one results in increased complexity and operational overhead, with a very low expected return on investment in the short to medium term while few end-users only have connectivity to the IPv6 Internet. Furthermore, the dual stack approach does not in any way help with the depletion of the IPv4 address space.

Therefore, some operators may prefer an approach in which they only need to operate one protocol in the data centre as they prepare for the future. The design goals are:

- Promote the deployment of native IPv6 services (cf. [RFC6540]).
- Provide IPv4 service availability for legacy users with no loss of performance or functionality.
- To ensure that that the legacy users’ IPv4 addresses remain available to the servers and applications.
- To conserve and maximise the utilisation of IPv4 addresses.
- To avoid introducing more complexity than absolutely necessary, especially on the servers and applications.
- To be easy to scale and deploy in a fault-tolerant manner.
The following subsections elaborate on how SIIT-DC meets these goals.

1.1.1. Single Stack IPv6 Operation

SIIT-DC allows an operator to build their applications on an IPv6-only foundation. IPv4 end-user connectivity becomes a service provided by the network, which systems administration and application development staff do not need to concern themselves with.

Obviously, this will promote universal IPv6 deployment for all of the provider’s services and applications.

It is worth noting that SIIT-DC requires no special support or change from the underlying IPv6 infrastructure, it will work with any kind of IPv6 network. Traffic between IPv6-enabled end users and IPv6-enabled services will always be native, and SIIT-DC will not be involved in it at all.

1.1.2. Stateless Operation

Unlike other solutions that provide either dual stack availability to single-stack services (e.g., Stateful NAT64 [RFC6146] and Layer-4/7 proxies), or that provide conservation of IPv4 addresses (e.g., NAPT44 [RFC3022]), a SIIT-DC Gateway does not keep any state between each packet in a single connection or flow. In this sense it operates exactly like a normal IP router, and has similar scaling properties - the limiting factors are packets per second and bandwidth. The number of concurrent flows and flow initiation rates are irrelevant for performance.

This not only allows individual SIIT-DC Gateways to easily attain "line rate" performance, it also allows for per-packet load balancing between multiple SIIT-DC Gateways using Equal-Cost Multipath Routing [RFC2991]. Asymmetric routing is also acceptable, which makes it easy to avoid sub-optimal traffic patterns; the prefixes involved may be anycasted from all the SIIT-DC Gateways in the provider’s network, thus ensuring that the most optimal path through the network is used, even where the optimal path in one direction differs from the optimal path in the opposite direction.

Finally, stateless operation means that high availability is easily achieved. If an SIIT-DC Gateway should fail, its traffic can be re-routed onto another SIIT-DC Gateway using a standard IP routing protocol. This does not impact existing flows any more than what any other IP re-routing event would.
1.1.3. IPv4 Address Conservation

In most parts of the world, it is difficult or even impossible to obtain generously sized IPv4 allocations from the Regional Internet Registries. The resulting scarcity in turn impacts individual end users and operators, which might be forced to purchase IPv4 addresses from other operators in order to cover their needs. This process can be risky to business continuity, in the case no suitable block for sale can be located, and/or turn out to be prohibitively expensive. Even so, a data centre operator will find that providing IPv4 service is essential, as a large share of the Internet users still does not have IPv6 connectivity.

A key goal of SIIT-DC is to help reduce a data centre operator’s IPv4 address requirement to the absolute minimum, by allowing the operator to remove them entirely from components that do not need to communicate with endpoints in the IPv4 Internet. One example would be servers that are operating in a supporting/back-end role and only communicates with to other servers (database servers, file servers, and so on). Another example would be the network infrastructure itself (router-to-router links, loopback addresses, and so on). Furthermore, as LAN prefix sizes must always be rounded up to the nearest power of two (or larger, if one reserves space for future growth), even more IPv4 addresses will often end up being wasted without even being used.

With SIIT-DC, the operator can remove these valuable IPv4 addresses from his back-end servers and network infrastructure, and reassign them to the SIIT-DC service as IPv4 Service Addresses. There is no requirement that IPv4 Service Addresses are assigned in an aggregated manner, so there is nothing lost due to infrastructure overhead; every single IPv4 address assigned to SIIT-DC can be used an IPv4 Service Address.

1.1.4. No Loss of End User’s IPv4 Source Address

SIIT-DC will map the entire end-user’s IPv4 source address into an predefined IPv6 translation prefix. This ensures that there is no loss of information; the end-user’s IPv4 source address remains available to the server/application, allowing it to perform tasks like Geo-Location, logging, abuse handling, and so forth.

1.1.5. Compatible with Standard IPv6 Implementations

Except for the introduction of the SIIT-DC Gateways themselves, no change to the network, servers, applications, or anything else is required in order to support SIIT-DC. SIIT-DC is practically invisible from the point of view of the the IPv4 clients, the IPv6...
servers, the IPv6 data centre network, and the IPv4 Internet. SIIT-DC interoperates with all standards-compliant IPv4 or IPv6 stacks.

1.1.6. No Architectural Dependency on IPv4

SIIT-DC will allow an ICP or data centre operator to build infrastructure and applications entirely on IPv6. This means that when the day comes to discontinue support for IPv4, no change needs to be made to the overall architecture - it’s only a matter of shutting off the SIIT-DC Gateways. Therefore, by deploying native IPv6 along with SIIT-DC, operators will avoid future migration or deployment projects relating to IPv6 roll-out and/or IPv4 sun-setting.

2. Terminology

This document makes use of the following terms:

IPv4 Service Address A public IPv4 address with which IPv4-only clients will communicate. This communication will be translated to IPv6 by the SIIT-DC Gateway.

IPv4 Service Address Pool One or more IPv4 prefixes routed to the SIIT-DC Gateway’s IPv4 interface. IPv4 Service Addresses are allocated from this pool. Note that this does not necessarily have to be a "pool" per se, as it could also be one or more host routes (whose prefix length is equal to /32). The primary purpose of using a pool rather than host routes is to facilitate IPv4 route aggregation and ease provisioning of new IPv4 Service Addresses.

IPv6 Service Address A public IPv6 address assigned to a server or application in the IPv6 network. IPv6-only and dual stacked clients communicates with this address directly without invoking SIIT-DC. IPv4-only clients also communicates with this address through the SIIT-DC Gateway and via an IPv4 Service Address.

SIIT-DC Host Agent A logical function very similar to an SIIT-DC Gateway that resides on a server and provides virtual IPv4 connectivity to applications, by reversing the translations done by the SIIT-DC Gateway. It is an optional component of the SIIT-DC architecture, that may be used to increase application support. See [I-D.anderson-v6ops-siit-dc-2xlat].

SIIT-DC Gateway A device or a logical function that translates between IPv4 and IPv6 in accordance with Section 5.
Static Address Mapping  A bi-directional mapping between an IPv4 Service Address and an IPv6 Service Address configured in the SIIT-DC Gateway. When translating between IPv4 and IPv6, the SIIT-DC Gateway changes the address fields in the translated packet’s IP header according to any matching Static Address Mapping.

Translation Prefix  An IPv6 prefix into which the entire IPv4 address space is mapped. This prefix is routed to the SIIT-DC Gateway’s IPv6 interface. It is either a Network-Specific Prefix or a Well-Known Prefix as specified in [RFC6052]. When translating between IPv4 and IPv6, the SIIT-DC Gateway prepends or strips the Translation Prefix from the address fields in the translated packet’s IP header, unless a Static Address Mapping exists for the IP address in question.

3. Architectural Overview

This section describes the basic SIIT-DC architecture.

SIIT-DC Architecture
Figure 1

In this example, 192.0.2.0/24 is allocated as an IPv4 Service Address Pool. Individual IPv4 Service Addresses are assigned from this pool. The provider must route this prefix to the SIIT-DC Gateway’s IPv4 interface. Note that there are no restrictions on how many IPv4 Service Address Pools are used or their prefix length, as long as they are all routed to the SIIT-DC Gateway’s IPv4 interface.
The Static Address Mapping list is used when translating an IPv4 Service Address (here 192.0.2.1) to its corresponding IPv6 Service Address (here 2001:db8:12:34::1) and vice versa. When the SIIT-DC Gateway translates an IPv4 packet to IPv6, any IPv4 Service Address found in the original IPv4 header will be replaced with the corresponding IPv6 Service Address in the resulting IPv6 header, and vice versa when translating an IPv6 packet to IPv4.

2001:db8:46::/96 is the Translation Prefix into which the entire IPv4 address space is mapped. It is used for translation of the end user’s IPv4 address to IPv6 and vice versa according to the algorithm defined in Section 2.2 of RFC6052 [RFC6052]. This algorithmic mapping has a lower precedence than the configured Static Address Mappings.

The SIIT-DC Gateway itself can be either a separate device or a logical function in another multi-purpose device, for example an IP router. Any number of SIIT-DC Gateways may exist simultaneously in an operators infrastructure, as long as they all have the same translation prefix and list of Static Mappings configured.

3.1. DNS Configuration

The IPv6 Service Address of should be registered in DNS using an "IN AAAA" record, while its corresponding IPv4 Service Address should be registered using an "IN A" record. This results in the following DNS records:

```
DNS Configuration for a SIIT-DC enabled service
www.example.com.    IN AAAA  2001:db8:12:34::1
www.example.com.    IN A     192.0.2.1
```

Figure 2

3.2. Packet Flow

In this example, "IPv4-only user" initiates a request to the application running on the IPv6-only server. He starts by looking up the "IN A" record of "www.example.org" in DNS, and attempts to connect to this address on the service by transmitting the following IPv4 packet destined for the IPv4 Service Address:

```
Stage 1: Client -> Server, IPv4
```
This packet is then routed over the Internet to the (nearest) SIIT-DC Gateway, which translates it into the following IPv6 packet and forward it into the IPv6 network:

Stage 2: Client -> Server request, IPv4

<table>
<thead>
<tr>
<th>IP Version:</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Address:</td>
<td>2001:db8:46::203.0.113.50</td>
</tr>
<tr>
<td>Destination Address:</td>
<td>2001:db8:12:34::1</td>
</tr>
<tr>
<td>Next Header:</td>
<td>TCP</td>
</tr>
<tr>
<td>TCP SYN [...]</td>
<td></td>
</tr>
</tbody>
</table>

The destination address field was translated to the IPv6 Service Address according to the configured Static Address Mapping, while the source address was field translated according to the [RFC6052] mapping using the Translation Prefix (because it did not match any Static Address Mapping). The rest of the IP header was translated according to [RFC6145]. The Layer 4 payload is copied verbatim, with the exception of the TCP checksum being recalculated.

Note that the IPv6 address 2001:db8:46::203.0.113.50 may also be expressed as 2001:db8:46::cb00:7132, cf. Section 2.2 of RFC4291 [RFC4291].

Next, the application receives receives this IPv6 packet and responds to it like it would with any other IPv6 packet:

Stage 3: Server -> Client response, IPv6
The response packet is routed to the (nearest) SIIT-DC Gateway’s IPv6 interface, which will translate it back to IPv4 as follows:

**Stage 4: Server -> Client response, IPv4**

```
+-------------------------------------------------+  
<p>| IP Version:           4                        |<br />
| Source Address:       192.0.2.2                |<br />
| Destination Address:  203.0.113.50             |</p>
<table>
<thead>
<tr>
<th>Protocol:             TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP SYN+ACK [...]</td>
</tr>
</tbody>
</table>
+-------------------------------------------------+  
```

Figure 6

This time, the source address matched the Static Address Mapping and was translated accordingly, while the destination address did not, and was therefore translated according to [RFC6052] by having the Translation Prefix stripped. The rest of the packet was translated according to [RFC6145].

The resulting IPv4 packet is transmitted back to the end user over the IPv4 Internet. Subsequent packets in the flow will follow the exact same translation pattern. They may or may not cross the same translators as earlier packets in the same flow.

The end user’s IPv4 stack has no idea that it is communicating with an IPv6 server, nor does the server’s IPv6 stack have any idea that is is communicating with an IPv4 client. To them, it’s just plain IPv4 or IPv6, respectively. However, the applications running on the server may optionally be updated to recognise and strip the Translation Prefix, so that the end user’s IPv4 address may be used for logging, Geo-Location, abuse handling, and so forth.

4. Deployment Guidelines
In this section, we list recommendations and guidelines for operators who would like to deploy a SIIT-DC service in their data centre network.

4.1. Application Support for NAT

Not all application protocols are able to operate in a network environment where rewriting of IP addresses occur. An operator should therefore carefully evaluate the applications he would like to make available for IPv4 users through SIIT-DC, to ensure they do not fall in this category. In general, if an application layer protocol works correctly through standard NAT44 (see [RFC3235]), it will most likely work correctly through SIIT-DC as well.

Higher-level protocols that embed IP addresses as part of their payload are especially problematic, as noted in [RFC2663], [RFC2993], and [RFC3022]. Such protocols will most likely not work through any form of address translation, including SIIT-DC. One well-known example of such a protocol is FTP [RFC0959].

The SIIT-DC architecture may be extended with a Host Agent that reverses the translation performed by the SIIT-DC Gateway before passing the packets to the application software. This allows the problematic application protocols described above to work correctly in an SIIT-DC environment as well. See [I-D.anderson-v6ops-siit-dc-2xlat] for a description of this extension.

4.2. Application Support for IPv6

SIIT-DC requires that the application software supports IPv6 networking, and that it has no dependency on IPv4 networking. If this is not the case, the approach described in [I-D.anderson-v6ops-siit-dc-2xlat] may be used, as it provides the application with seemingly native IPv4 connectivity. This allows IPv4-only applications to work correctly in an otherwise IPv6-only environment.

4.3. Application Communication Pattern

SIIT-DC is ideally suited for applications where IPv4-only nodes on the Internet initiate traffic towards the IPv6-only services, which in turn are only passively listening for inbound traffic and responding as necessary. One well-known example of such a protocol is HTTP [RFC7230]. This is due to the fact that in this case, an IPv4 user looks exactly like an ordinary IPv6 user from the host and application’s point of view, and requires no special treatment.
It is possible to combine SIIT-DC with DNS64 [RFC6147] in order to allow an IPv6-only application to initiate communication with IPv4-only nodes through an SIIT-DC Gateway. However, in this case, care must be taken so that all outgoing communication is sourced from the IPv6 Service Address that has a Static Mapping configured on the SIIT-DC Gateway. If another unmapped address is used, the SIIT-DC Gateway will discard the packet.

An alternative approach to the above would be to make use of an SIIT-DC Host Agent as described in [I-D.anderson-v6ops-siit-dc-2xlat]. This provides the application with seemingly native IPv4 connectivity, which it may use for both inbound and outbound communication without requiring the application to select a specific source address for its outbound communications.

4.4. Choice of Translation Prefix

Either a Network-Specific Prefix (NSP) from the provider’s own IPv6 address space or the IANA-allocated Well-Known Prefix 64:ff9b::/96 (WKP) may be used. From a technical point of view, both should work equally well, however as only a single WKP exists, if a provider would like to deploy more than one instance of SIIT-DC in his network, or Stateful NAT64 [RFC6146], an NSP must be used anyway for all but one of those deployments.

Furthermore, the WKP cannot be used in inter-domain routing. By using an NSP, a provider will have the possibility to provide SIIT-DC service to other operators across Autonomous System borders.

For these reasons, this document recommends that an NSP is used. Section 3.3 of [RFC6052] discusses the choice of translation prefix in more detail.

The Translation Prefix may use any of the lengths described in Section 2.2 of RFC6052 [RFC6052], but /96 has two distinct advantages over the others. First, converting it to IPv4 can be done in a single operation by simply stripping off the first 96 bits; second, it allows for IPv4 addresses to be embedded directly into the text representation of an IPv6 address using the familiar dotted quad notation, e.g., "2001:db8::198.51.100.10" (cf. Section 2.4 of RFC6052 [RFC6052])), instead of being converted to hexadecimal notation. This makes it easier to write IPv6 ACLs and similar that match translated endpoints in the IPv4 Internet. Use of a /96 prefix length is therefore recommended.

4.5. Routing Considerations
The prefixes that constitute the IPv4 Service Address Pool and the IPv6 Translation Prefix may be routed to the SIIT-DC Gateway(s) as any other IPv4 or IPv6 route in the provider’s network.

If more than one SIIT-DC Gateway is being deployed, it is recommended that a dynamic routing protocol (such as BGP, IS-IS, or OSPF) is being used to advertise the routes within the provider’s network. This will ensure that the traffic that is to be translated will reach the closest SIIT-DC Gateway, reducing or eliminating sub-optimal traffic patterns, as well as provide high availability – if one SIIT-DC Gateway fails, the dynamic routing protocol will automatically redirect the traffic to the next-best translator.

4.6. Location of the SIIT-DC Gateways

The goal of SIIT-DC is to facilitate a true IPv6-only application and network architecture, with the sole exception being the IPv4 interfaces of the SIIT-DC Gateways and the network infrastructure required to connect them to the IPv4 Internet. Therefore, the SIIT-DC Gateways should be located somewhere between the IPv4 Internet and the application delivery stack. This should be understood to include all servers, load balancers, firewalls, intrusion detection systems, and similar devices that are processing traffic to a greater extent than merely forwarding it.

It is optimal to place the SIIT-DC Gateways as close as possible to the direct path between the servers and the end users. If the closest translator is located a long way from the optimal path, all packets in both directions must make a detour. This would increase the RTT between the server and the end user by by two times the extra latency incurred by the detour, as well as cause unnecessary load on the network links on the detour path.

Where possible, it is beneficial to implement the SIIT-DC Gateways as a logical function within the routers would have handled the traffic anyway, had the topology been dual stacked. This way, the translation service would not need to be assigned separate networks ports (which might become saturated and impact the service quality), nor would it require extra rack space and energy. Some particularly good choices of the location could be within a data centre’s access routers, or within the provider’s border routers. When every single application in the data centre or the provider’s network eventually runs on single-stack IPv6, there would no need to run IPv4 on the inside of the SIIT-DC Gateway. This reduces complexity, and allows the operator to reclaim IPv4 addresses from the network infrastructure that may instead be used as IPv4 Service Address Pools.
Finally, another possibility is that the data centre operator outsources the SIIT-DC service to another entity, for example his upstream ISP. Doing so allows the data centre operator to build a true IPv6-only infrastructure. However, in this case, care must be taken to ensure that the path between the data centre and the SIIT-DC operator has a stable and known MTU, and that the SIIT-DC Gateways are not too far away from the data centre (otherwise, translated traffic could incur a latency penalty).

4.7. Migration from Dual Stack

While this document discusses the use of IPv6-only servers and applications, there is no technical requirement that the servers are IPv4 free. SIIT-DC works equally well for dual stacked servers, which makes migration easy - after setting up the translation function, the DNS "IN A" record for the service is updated to point to the IPv4 address that will be translated to IPv6, the previously used IPv4 service address may continue to be assigned to the server. This makes roll-back to dual stack easy, as it is only a matter of changing the DNS record back to what it was before.

It is also possible to use DNS Round Robin to gradually migrate a dual-stacked service’s IPv4 traffic from native to SIIT-DC. This is done by configuring multiple DNS "IN A" records for the service’s hostname, and pointing one portion of them to the service’s native IPv4 addresses and another portion to IPv4 Service Addresses handled by SIIT-DC. The distribution of "IN A" records determines how much of the client traffic will pass through the SIIT-DC Gateway and how much will remain native. This operator may then gradually increase the share of SIIT-DC "IN A" DNS records until no native addresses remain.

When all client traffic is handled by SIIT-DC, the operator may proceed to remove the (now unused) IPv4 addresses assigned to the servers in question. They could then potentially be recycled as another IPv4 Service Address Pool assigned to SIIT-DC.

4.8. Packet Size and Fragmentation Considerations

There are some key differences between IPv4 and IPv6 relating to packet sizes and fragmentation that one should consider when deploying SIIT-DC. They result in a few problematic corner cases, which can be dealt with in a few different ways. The following subsections will discuss these in detail, and provide operational guidance.

In particular, the operator may find that relying on fragmentation in the IPv6 domain is undesired or even operationally impossible.
For this reason, the recommendations in this section seek to minimise the use of IPv6 fragmentation. Unless otherwise stated, the following subsections assume that the MTU in both the IPv4 and IPv6 domains is 1500 bytes.

4.8.1. IPv4/IPv6 Header Size Difference

The IPv6 header is up to 20 bytes larger than the IPv4 header. This means that a full-size 1500 bytes large IPv4 packet cannot be translated to IPv6 without being fragmented, otherwise it would likely have resulted in a 1520 bytes large IPv6 packet.

If the transport protocol used is TCP, this is generally not a problem, as the IPv6 server will advertise a TCP MSS of 1440 bytes. This causes the client to never send larger packets than what can be translated to a single full-size IPv6 packet, eliminating any need for fragmentation.

For other transport protocols, full-size IPv4 packets with the DF flag cleared will need to be fragmented by the SIIT-DC Gateway. The only way to avoid this is to increase the Path MTU between the SIIT-DC Gateway and the servers to 1520 bytes. Note that the servers’ MTU SHOULD NOT be increased accordingly, as that would cause them to undergo Path MTU Discovery for most native IPv6 destinations. However, the servers would need to be able to accept and process incoming packets larger than their own MTU. If the server’s IPv6 implementation allows the MTU to be set differently for specific destinations, it could be increased to 1520 for destinations within the Translation Prefix specifically.

4.8.2. IPv6 Atomic Fragments

In keeping with the fifth paragraph of Section 4 of RFC6145 [RFC6145], an SIIT-DC Gateway will by default add an IPv6 Fragmentation header to the resulting IPv6 packet when translating an IPv4 packet with the Don’t Fragment flag set to 0.

This happens even though the resulting IPv6 packet isn’t actually fragmented into several pieces, resulting in an IPv6 Atomic Fragment [RFC6946]. These Atomic Fragments are generally not useful in a data centre environment, and it is therefore recommended that this behaviour is disabled in the SIIT-DC Gateways. To this end, Section 4 of RFC6145 [RFC6145] notes that the "translator MAY provide a configuration function that allows the translator not to include the Fragment Header for the non-fragmented IPv6 packets".
Note that [I-D.ietf-6man-deprecate-atomfrag-generation] seeks to update [RFC6145], making the functionality described above as the standard and only mode of operation.

In IPv6, the Identification value is located inside the Fragmentation header. That means that if the generation of IPv6 Atomic Fragments is disabled, the IPv4 Identification value will be lost during translation to IPv6. This could potentially confuse some diagnostic tools.

4.8.3. Minimum Path MTU Difference Between IPv4 and IPv6

Section 5 of RFC2460 [RFC2460] specifies that the minimum IPv6 link MTU is 1280 bytes. Therefore, an IPv6 node can reasonably assume that if it transmits an IPv6 packet that is 1280 bytes or smaller, it is guaranteed to reach its destination without requiring fragmentation or invoking the Path MTU Discovery algorithm [RFC1981]. However, this assumption fails if the destination is an IPv4 node reached through a protocol translator such as an SIIT-DC Gateway, as the minimum IPv4 link MTU is 68 bytes. See Section 3.2 of RFC791 [RFC0791].

Section 5.1 of RFC6145 [RFC6145] specifies that an SIIT-DC Gateway should set the IPv4 Don’t Fragment flag to 1 when it translates a non-fragmented IPv6 packet to IPv4. This means that when the path to the destination IPv4 node contains an IPv4 link with an MTU smaller than 1260 bytes (which corresponds to an IPv6 MTU smaller than 1280 bytes, cf. Section 4.8.1), the Path MTU Discovery algorithm will be invoked, even if the original IPv6 packet was only 1280 bytes large. This happens as a result of the IPv4 router connecting to the IPv4 link with the small MTU returning an ICMPv4 Need To Fragment error with an MTU value smaller than 1260, which in turns is translated by the SIIT-DC Gateway to an ICMPv6 Packet Too Big error with an MTU value smaller than 1280 which is then transmitted to the origin IPv6 node.

When an IPv6 node receives an ICMPv6 Packet Too Big error indicating an MTU value smaller than 1280, the last paragraph of Section 5 of RFC2460 [RFC2460] gives it two choices on how to proceed:

- It may reduce its Path MTU value to the value indicated in the Packet Too Big, i.e., limit the size of subsequent packets transmitted to that destination to the indicated value. This approach causes no problems for the SIIT-DC function, as it simply allows Path MTU Discovery to work transparently across the SIIT-DC Gateway.
It may reduce its Path MTU value to exactly 1280, and in addition include a Fragmentation header in subsequent packets sent to that destination. In other words, the IPv6 node will start emitting Atomic Fragments. The Fragmentation header signals to the the SIIT-DC Gateway that the Don’t Fragment flag should be set to 0 in the resulting IPv4 packet, and it also provides the Identification value.

If the use of the IPv6 Fragmentation header is problematic, and the operator has IPv6 nodes that implement the second option above, the operator should consider enabling the functionality described as the "second approach" in Section 6 of RFC6145 [RFC6145]. This functionality changes the SIIT-DC Gateway’s behaviour as follows:

- When translating ICMPv4 Need To Fragment to ICMPv6 Packet Too Big, the resulting packet will never contain an MTU value lower than 1280. This prevents the IPv6 nodes from generating Atomic Fragments.

- When translating IPv6 packets smaller than or equal to 1280 bytes, the Don’t Fragment flag in the resulting IPv4 packet will be set to 0. This ensures that in the eventuality that the path contains an IPv4 link with an MTU smaller than 1260, the IPv4 router connected to that link will have the responsibility to fragment the packet before forwarding it towards its destination.

In summary, this approach could be seen as prompting the IPv4 protocol itself to provide the "link-specific fragmentation and reassembly at a layer below IPv6" required for links that "cannot convey a 1280-octet packet in one piece", to paraphrase Section 5 of RFC2460 [RFC2460]. Note that [I-D.ietf-6man-deprecate-atomfrag-generation] seeks to update [RFC6145], making the approach described above as the standard and only mode of operation.

5. Implementation Requirements

This normative section specifies the SIIT-DC protocol that is implemented by an SIIT-DC Gateway. Because SIIT-DC builds on and closely resembles SIIT [RFC6145], this section should be read as a set of additions and changes that are applied to an implementation already compliant to SIIT [RFC6145]. Each of the following subsections discuss how the requirement relates to with any corresponding requirements in SIIT [RFC6145].

5.1. Compliance with RFC6145 and RFC6052
Unless otherwise stated in the following sections, an SIIT-DC implementation MUST comply fully with [RFC6145]. It must also implement the algorithmic address mapping defined in [RFC6052].

5.2. Static Address Mapping Function

The implementation MUST allow the operator to configure an arbitrary number of Static Address Mappings which override the default [RFC6052] algorithm. It SHOULD be possible to specify a single bi-directional mapping that will be used in both the IPv4=>IPv6 and IPv6=>IPv4 directions, but it MAY additionally (or alternatively) support unidirectional mappings.

An example of such a bidirectional Static Address Mapping would be:

- 192.0.2.1 <=> 2001:db8:12:34::1

To accomplish the same using unidirectional mappings, the following two mappings must instead be configured:

- 192.0.2.1 => 2001:db8:12:34::1
- 2001:db8:12:34::1 => 192.0.2.1

In both cases, if the SIIT-DC Gateway receives an IPv6 packet that has the value 2001:db8:12:34::1 in either the source or destination field of the IPv6 header, it MUST rewrite this field to 192.0.2.1 when translating to IPv4. Similarly, if the SIIT-DC Gateway receives an IPv4 packet that has the value 192.0.2.1 as the either the source or destination field of the IPv4 header, it MUST rewrite this field to 2001:db8:12:34::1 when translating to IPv6. For all IPv4 or IPv6 source or destination field values for which there are no matching Static Address Mapping, [RFC6052] compliant mapping MUST be used instead.

Relation to [RFC6145]: The Static Address Mapping is a novel feature feature that is not discussed in [RFC6145]. It conflicts with the [RFC6145] requirement that all addresses must be translated according to the [RFC6052] algorithm.

5.3. Support for Increasing the IPv6 Path MTU

The SIIT-DC Gateway 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 resulting IPv6 packets.
Relation to [RFC6145]: This strengthens the corresponding "MAY" requirement located in Section 4 of RFC6145 [RFC6145] to a "MUST".

5.4.  Loop Prevention Mechanism

As noted in Section 9.2, there is a potential for packets looping through the SIIT-DC function if it receives an IPv4 packet for which there is no Static Address Mapping. It is therefore RECOMMENDED that the implementation has a mechanism that automatically prevents this behaviour. One way this could be accomplished would be to discard any IPv4 packets that would be translated into an IPv6 packet that would be routed straight back into the SIIT-DC function.

If such a mechanism isn’t provided, the implementation MUST provide a way to manually filter or null-route the destination addresses that would otherwise cause loops.

Relation to [RFC6145]: This security consideration applies only when an SIIT-DC Gateway translates a packet in "pure" SIIT [RFC6145] mode (i.e., when both address fields are translated according to [RFC6052]). This consideration is in other words not specific to SIIT-DC, it is inherited from [RFC6145]. In spite of this, [RFC6145] does not describe this consideration or any methods of prevention. The requirements in this section is therefore novel to SIIT-DC, even though they apply equally to [RFC6145].

6.  Acknowledgements

The author would like to thank the following individuals for their contributions, suggestions, corrections, and criticisms: Fred Baker, Cameron Byrne, Brian E Carpenter, Ross Chandler, Dagfinn Ilmari Mannsaaker, Lars Olafsen, Stig Sandbeck Mathisen, Knut A. Syed, Andrew Yourtchenko.

7.  Requirements Language

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].

8.  IANA Considerations

This draft makes no request of the IANA. The RFC Editor may remove this section prior to publication.

9.  Security Considerations
9.1. Mistaking the Translation Prefix for a Trusted Network

If a Network-Specific Prefix from the provider’s own address space is chosen for the translation prefix, as is recommended, care must be taken if the translation service is used in front of services that have application-level ACLs that distinguish between the operator’s own networks and the Internet at large, as the translated IPv4 end users on the Internet will appear to be located within the provider’s own IPv6 address space. It is therefore important that the translation prefix is treated the same as the Internet at large, rather than as a trusted network.

9.2. Packets Looping Through the SIIT-DC Function

If the SIIT-DC Gateway receives an IPv4 packet destined to an address for which there is no Static Address Mapping, its destination address will be rewritten according to [RFC6052], making the resulting IPv6 packet have a destination address within the translation prefix, which is likely routed to back to the SIIT-DC function. This will cause the packet to loop until its Time To Live / Hop Limit reaches zero, potentially creating a Denial Of Service vulnerability.

To avoid this, it should be ensured that packets sent to IPv4 destinations addresses for which there are no Static Address Mappings, or whose resulting IPv6 address does not have a more-specific route to the IPv6 network, are immediately discarded.

10. References

10.1. Normative References


10.2. Informative References

[I-D.ietf-6man-deprecate-atomfrag-generation]

[I-D.taylor-v6ops-fragdrop]


Appendix A. Complete SIIT-DC topology example

This figure shows a more complete SIIT-DC topology, in order to better demonstrate the beneficial properties it has. In particular, it tries to highlight the following:

- Stateless operation: Any number of SIIT-DC Gateways may be deployed side-by-side, or indeed anywhere in the IPv6 network, as any standard routing mechanism may be used to direct traffic to them (shown here with BGP on the IPv4 side and ECMP on the IPv6 side). This in turn leads to high availability, should one of the SIIT-DC Gateways fail or become unavailable, those standard routing mechanisms will ensure that traffic is automatically redirect one of the remaining SIIT-DC Gateways.
IPv4 address conservation: Even though the to customers in the example have several hundred servers, most of them are not used for externally available services, and thus do not require an IPv4 address. The network between the servers and the SIIT-DC Gateways require no IPv4 addresses, either. Furthermore, the IPv4 addresses that are used do not have to be assigned to customers in the form of aggregated blocks or prefixes; which makes it easy to achieve 100% effective utilisation of the IPv4 service address pools.

Application support: The translation-friendly applications HTTP and SMTP will work through SIIT-DC without requiring any special customisation. Furthermore, translation-unfriendly applications such as FTP will also work if an host agent in present, cf. [I-D.anderson-v6ops-siit-dc-2xlat].

Native IPv6 as the foundation: Every server, application, and network component has access to native and untranslated IPv6 connectivity to each other and to the Internet. Traffic through the SIIT-DC Gateways will diminish over time as IPv6 is deployed throughout the Internet. Eventually they may be shut down entirely, which causes no disruption to the application stacks’ ability to deliver their services over native IPv6.
IPv4 Internet <-> [BGP] <-> IPv6 Internet

---<192.0.2.0/24>---

<table>
<thead>
<tr>
<th>SIIT-DC Gateway 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation Prefix:</td>
</tr>
<tr>
<td>2001:db8:46::/96</td>
</tr>
</tbody>
</table>

Static Address Mappings:
192.0.2.1 <-> 2001:db8:12:34::1
192.0.2.2 <-> 2001:db8:12:34::2
192.0.2.3 <-> 2001:db8:fe:dc::1
192.0.2.4 <-> 2001:db8:12:34::4
[...]

---<2001:db8:46::/96>---

---[ECMP]---

Customer A’s server LAN
2001:db8:12:34::/64

--- www ::1 (IPv6+SIIT-DC) ---
+-- file01 ::f:01 (IPv6) +-- file99 ::f:99 (IPv6)

+-- mta ::2 (IPv6+SIIT-DC) +-- [...]

+-- ftp ::3 (IPv6)
  +-- ::4 (SIIT-DC/Host Agent)

--- app01 ::a:01 (IPv6) ---
+-- app99 ::a:99 (IPv6)

--- db01 ::d:01 (IPv6) ---
+-- db99 ::d:99 (IPv6)

Customer B’s server LAN
2001:db8:fe:dc::/64

--- www ::1 (IPv6+SIIT-DC) ---

--- mta ::2 (IPv6+SIIT-DC) ---

--- ftp ::3 (IPv6) ---

--- app01 ::a:01 (IPv6) ---
+-- app99 ::a:99 (IPv6)

--- db01 ::d:01 (IPv6) ---
+-- db99 ::d:99 (IPv6)
Appendix B. Comparison to Other Deployment Approaches

There are a number of alternative deployment strategies a data centre operator may follow. They each have different properties and help solve a different set of challenges. This section aims to compare the SIIT-DC approach with each of the most common ones, by highlighting the benefits and disadvantages of each.

B.1. IPv4-only

At the time of writing, IPv4-only operation remains the status quo for most operators. As such, it is well understood and supported. An operator can reasonably expect everything to work correctly in an IPv4-only environment.

Benefits of IPv4-only operation compared to SIIT-DC include:
- No translation occurs, the end-to-end principle is intact.
- Compatible with all common application protocols.
- Compatible with IPv4-only devices.
- Compatible with IPv4-only application software, without requiring a host agent.

Disadvantages of IPv4-only operation compared to SIIT-DC include:
- Does not provide any form of IPv6 connectivity.
- Does not alleviate IPv4 address scarcity.

B.2. IPv4-only + NAPT44

An operator who would otherwise chose a traditional IPv4-only approach, but cannot due to having insufficient public IPv4 addresses available, could chose to deploy using a combination of private IPv4 addresses [RFC1918] and NAPT44 [RFC3022] devices which will translate between a smaller number of public IPv4 addresses and the private addresses assigned to the servers that provide public services to the Internet.

Benefits of IPv4-only + NAPT44 operation compared to SIIT-DC include:
- Compatible with IPv4-only devices.
Compatible with IPv4-only application software, without requiring a host agent.

Disadvantages of IPv4-only + NAPT44 operation compared to SIIT-DC include:

- Does not provide any form of IPv6 availability.
- Requires network devices that track all flow state, which may create a performance bottleneck and be an easy target for Denial of Service attacks.
- Limits routing flexibility (prevents closest exit routing), as outbound traffic must pass across the same NAPT44 device that handled the inbound traffic.
- Limited potential for horizontal scaling, as packets cannot be load-balanced across multiple NAT devices.
- Depending on whether or not the NAPT44 device rewrites source addresses in order to attract the return traffic to itself:
  
  * Obscures the true source address of the user from the server/application, preventing it from e.g. performing geo-location lookups, or:
  * Requires an IPv4 default route to be pointed to the NAPT44 device, also attracting native traffic that does not need to undergo translation.

In addition, application compatibility is a consideration with both NAPT44 and SIIT-DC, but the exact nature depends from application to application, so it is hard to objectively quantify if there is a clear advantage to either approach here. Some translation-unfriendly application protocols may work without host modifications through the use of Application Layer Gateway support in the NAPT44 device (e.g., FTP [RFC0959]), or in the SIIT-DC architecture when a host agent is being used [I-D.anderson-v6ops-siit-dc-2xlat]. Other application protocols might not work with NAPT44 at all, but will work in the SIIT-DC if a host agent is being used (e.g., FTP/ILS [RFC4217]).

In summary, the most accurate statement would be to say that an NAPT44 architecture is more compatible with translation-unfriendly protocols than plain SIIT-DC, while SIIT-DC is more compatible than NAPT44 if a host agent is used.
For a more complete discussion of potential issues with running NAPT44, see Architectural Implications of NAT [RFC2993].

B.3. IPv4-only + NAT64

An operator who would otherwise chose a traditional IPv4-only approach, but would in addition like to provide service availability for IPv6 end users, could use Stateful NAT64 [RFC6146] to accomplish this. In a sense, this would be the mirror image of an SIIT-DC architecture: The infrastructure and servers remains single-stacked, while connectivity to the other IP stack is provided through a translation system. Further information about operating Stateful NAT64 is found in [RFC7269].

Note that Stateful NAT64 can be deployed with or without NAPT44. With the exception that IPv6 service availability is being provided, the discussion in the previous two sections fully applies to an IPv4-only environment that includes NAT64.

Benefits of IPv4-only + NAT64 operation compared to SIIT-DC include:

- Compatible with IPv4-only devices.
- Compatible with IPv4-only application software, without requiring a host agent.

Disadvantages of IPv4-only + NAT64 operation compared to SIIT-DC include:

- Does not alleviate IPv4 address scarcity (assuming NAPT44 isn’t used).
- Requires network devices that track all flow state, which may create a performance bottleneck and be an easy target for Denial of Service attacks.
- Limits routing flexibility (prevents closest exit routing), as outbound traffic must pass across the same NAT64 device that handled the inbound traffic.
- Limited potential for horizontal scaling, as packets cannot be load-balanced across multiple NAT devices.
- Obscures the true source address of the user from the server/application, preventing it from e.g. performing geo-location lookups.
The traffic levels on the Stateful NAT64 routers will increase over time, in lockstep with the increased deployment of IPv6 in the Internet. For this reason, Section 3.2 of RFC7269 [RFC7269] notes that the use of Stateful NAT64 in a data centre environment "is only reasonable at an early stage". With SIIT-DC, the inverse is true; the traffic levels on the SIIT-DC Gateways will decrease over time, as end users will prefer to use native IPv6 once it is available to them.

B.4. Dual Stack

Dual Stack [RFC4213] [RFC6883] could be used both with or without NAPT44 to handle IPv4. In general, the benefits and disadvantages are equal to the corresponding IPv4-only option, except for the fact that Dual Stack does provide IPv6 connectivity. Therefore, his section only lists the benefits and disadvantages which are unique to a Dual Stack environment.

Benefits of Dual Stack operation compared to SIIT-DC include:

- No translation occurring, the end-to-end principle is intact (assuming NAPT44 isn't used).
- Compatible with all common application protocols (assuming NAPT44 isn't used).
- Compatible with IPv4-only devices.
- Compatible with IPv4-only application software, without requiring a host agent.

Disadvantages of Dual Stack operation compared to SIIT-DC include:

- Does not alleviate IPv4 address scarcity (assuming NAPT44 isn’t used).
- Increases the complexity of the infrastructure, as many things must done twice (once for IPv4 and once for IPv6). Examples of things that must be duplicated in this manner under Dual Stack include: Firewall rules/ACLs, IGP topology, monitoring, troubleshooting.
- Encourages software developers, systems administrators, etc. to build architectures that cannot operate correctly without IPv4. This in turn makes it difficult to make use of Dual Stack as a short term transitional stage, rather than a near-permanent end state.
o Increases the amount of things that can encounter failures, and increases the time required to locate and fix such failures. This reduces reliability.

B.5. Partial Dual Stack (IPv6-only back-end)

It is possible to use the Dual Stack deployment strategy for front-end services only. That is, the front-end servers (or load balancers) that serve public Internet-available services are provisioned with both native IPv4 and native IPv6 connectivity on their Internet-facing interfaces, while the interfaces facing the back-end infrastructure are IPv6 only. All back-end servers that do not communicate directly with Internet clients are IPv6-only. All communication between back-end servers as well as all traffic between the back-end servers and the front-end servers will therefore use only IPv6.

One variation of this approach is to have two separate sets of front-end servers, where one set has IPv4-only Internet-facing interfaces, while the other set has IPv6-only Internet-facing interfaces. However, both sets must have IPv6-only interfaces facing the back-end infrastructure.

Benefits of Partial Dual Stack operation compared to SIIT-DC include:

o No translation occurring, the end-to-end principle is intact.

o Compatible with all common application protocols.

o Compatible with IPv4-only devices (front-end only).

o Compatible with IPv4-only application software (front-end only).

Disadvantages of Partial Dual Stack operation compared to SIIT-DC include:

o Increases the complexity of the front-end infrastructure, as many things must be done twice (once for IPv4 and once for IPv6). Examples of things that must be duplicated in this manner under Partial Dual Stack include: firewall rules/ACLs, IGP topology, monitoring, troubleshooting.

o Can not support any IPv4-only devices or application software in the back-end infrastructure.

In addition, Partial Dual Stack will alleviate IPv4 address scarcity compared to the normal Dual Stack approach, but not quite to the same extent as SIIT-DC. This is primarily due to the data centre network...
infrastructure having to be dual-stacked in order to provide native IPv4 addressing to the front-end servers, and because the front-end server LANs must be rounded up in size to the nearest CIDR boundary which may result in IPv4 addresses being unused. However, depending on the exact circumstances, this difference in IPv4 address consumption between SIIT-DC and Partial Dual Stack may be negligible.

Author’s Address

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
SIIT-DC: Dual Translation Mode
draft-ietf-v6ops-siit-dc-2xlat-00

Abstract

This document describes an extension of the Stateless IP/ICMP Translation for IPv6 Data Centre Environments architecture (SIIT-DC), which allows applications, protocols, or nodes that are incompatible with IPv6, SIIT-DC and/or Network Address Translation in general to operate correctly in an SIIT-DC environment. This is accomplished by introducing a new component called an Edge Translator, which reverses the translations made by an SIIT-DC Gateway. The application or device is thus provided with seemingly native IPv4 connectivity.

The reader is expected to be familiar with the SIIT-DC architecture described in I-D.ietf-v6ops-siit-dc.

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 July 29, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of
1. Introduction

SIIT-DC [I-D.ietf-v6ops-siit-dc] describes an architecture where IPv4-only users can access IPv6-only services through a stateless translator called an SIIT-DC Gateway. This approach has certain limitations, however. In particular, the following cases will work poorly or not at all:

- Application protocols that do not support NAT (i.e., the lack of end-to-end transparency of IP addresses).
- Devices which cannot connect to IPv6 networks at all, or which can only connect such networks if they also provide IPv4 connectivity (i.e., dual-stacked networks).
- Application software which makes use of legacy IPv4-only APIs, or otherwise makes assumptions that IPv4 connectivity is available.
By extending the SIIT-DC architecture with a new component called an Edge Translator (ET), all of the above can be made to work correctly in an otherwise IPv6-only network environment using SIIT-DC.

The purpose of the Edge Translator is to reverse the IPv4-to-IPv6 packet translations previously done by the SIIT-DC Gateway for traffic arriving from IPv4 clients and forward this as "native" IPv4 to the application software or device. In the reverse direction, IPv4 packets transmitted by the application software or device is intercepted by the Edge Translator, which will translate them to IPv6 before they are forwarded to the SIIT-DC Gateway, which in turn will reverse the translations and forward them to the IPv4 End User. In short, the device or application software is provided with "virtual" IPv4 Internet connectivity that retains end-to-end transparency for the IPv4 addresses.

2. Terminology

This document makes use of the following terms:

Edge Translator (ET)
A device or logical function that provides "native" IPv4 connectivity to IPv4-only devices or application software. It is very similar in function to an SIIT-DC Gateway, but is typically located close to the IPv4-only component(s) it is supporting rather than on the network border.

IPv4 Service Address
A public IPv4 address with which IPv4-only clients will communicate. This communication will be translated to IPv6 by the SIIT-DC Gateway and back to IPv4 again by the Edge Translator.

SIIT-DC Gateway
A device or a logical function that translates between IPv4 and IPv6 in accordance with [I-D.ietf-v6ops-siit-dc].

Static Address Mapping
A bi-directional mapping between an IPv4 Service Address and an IPv6 Service Address configured in the SIIT-DC Gateway. When translating between IPv4 and IPv6, the SIIT-DC Gateway changes the address fields in the translated packet’s IP header according to any matching Static Address Mapping.

Translation Prefix
An IPv6 prefix into which the entire IPv4 address space is mapped. This prefix is routed to the SIIT-DC Gateway’s IPv6 interface. It is either an Network-Specific Prefix or a Well-Known Prefix as specified in [RFC6052]. When translating between IPv4 and IPv6,
the SIIT-DC Gateway will prepend or strip the Translation Prefix from the address fields in the translated packet’s IP header, unless a Static Address Mapping exists for the IP address in question.

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

3. Edge Translator Description

An Edge Translator (ET) is at its core an implementation of the Stateless IP/ICMP Translation algorithm [RFC6145], with the Static Address Mapping extension described in Section 5.2 of [I-D.ietf-v6ops-siit-dc]. It provides virtual IPv4 connectivity for application software or devices which require this to operate correctly in an SIIT-DC environment.

Inbound IPv4 packets destined for an IPv4 Service Address is first translated to IPv6 by an SIIT-DC Gateway. The resulting IPv6 packets are subsequently forwarded to the ET handling the IPv6 Service Address they are addressed to. The ET then translates them back to IPv4 before forwarding them to the IPv4 application software or device. In the other direction, the exact same translations happen, only in reverse. This process provides end-to-end transparency of IPv4 addresses.

An ET may handle an arbitrary number of IPv4 Service Addresses. All the Static Address Mappings configured in the SIIT-DC Gateway(s) that involve the IPv4 Service Addresses handled by an ET MUST be duplicated in that ET’s configuration.

An ET may be implemented in two distinct ways; as a software-based service residing inside an otherwise IPv6-only host, or as a network-based service that provides an isolated IPv4 network segment to which devices which require IPv4 can connect. In both cases native IPv6 connectivity may be provided simultaneously with the virtual IPv4 connectivity. Thus, dual-stack connectivity is facilitated in case the device or application software support it.

The choice between a host- or network-based ET is made on a per-service or -device basis. An arbitrary number of each type of ET may co-exist in an SIIT-DC architecture.
This section describes the different approaches and discusses which approach fits best for the various use cases.

3.1. Host-Based Edge Translator

Overview of a Host-based Edge Translator

```
+--------<SIIT-DC GW>--+
|                   |      
+---|--[XLAT]|---+
| [IPv4 Internet] | [IPv6 Internet] |
+--------<IPv6-only server>--------+
|                   |      
+---+[ET/XLAT]--AF_INET Dual-stack+  
|                     | Application |
+-------------------+            
|                   |      
+-------------------+            
```

Figure 1

A host-based Edge Translator is typically implemented as a logical software function that runs inside the operating system of a host or server. It provides software applications running on the same host with IPv4 connectivity. The IPv4 Service Address it handles is considered local, allowing application software running on the same host to use traditional IPv4-only API calls, e.g., to create AF_INET sockets that listens for and accepts incoming connections to its IPv4 Service Address. An ET could accomplish this by creating an virtual network adapter to which it assigns the IPv4 Service Address and points a default IPv4 route.

As shown in Figure 1, if the application software supports dual-stack operation, IPv6 clients will be able to communicate with it directly using native IPv6. Neither the SIIT-DC Gateway nor the ET will intercept this communication. Support for IPv6 in the application software is however not a requirement; the application software may opt not to establish any IPv6 sockets. Foregoing IPv6 in this manner will simply preclude connectivity to the service from IPv6-only clients; connectivity to the service from IPv4 clients (through the SIIT-DC Gateway) will work in the exact same manner in both cases.

The ET requires a dedicated IPv6 Service Address for each IPv4 Service Address it has configured. The IPv6 network must forward
traffic to these IPv6 Service Addresses to the host, whose operating system must in turn forward them to the ET. This document does not explore the multitude of ways this could be accomplished, however considering that the IPv6 protocol is designed for having multiple addresses assigned to a single node, one particularly straight-forward way would be to assign the ET’s IPv6 Service Addresses as secondary IPv6 addresses on the host itself so that it the upstream router learns of their location using the IPv6 Neighbor Discovery Protocol [RFC4861].

3.2. Network-Based Edge Translator

Overview of a Basic Network-based Edge Translator

```
[IPv4 Internet]   [IPv6 Internet]  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>--&lt;-SIIT-DC GW-&gt;---</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>+----------------+</td>
<td></td>
</tr>
<tr>
<td>[XLAT]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
+----------------+    |
|                |    |
| [IPv6-only data centre network] |
|    |    |
+----------------+    |
|                 |    |
| --<-ET>---+    |
| [XLAT]        |
| +-----------+ |
|              |
+----------------+ |
|               |
| [Isolated IPv4-only network segment] |
|    |    |
| +----------------+    |
|                  |    |
| --<-IPv4-only server>----+  |
| +----------------+    |
| \--AF_INET IPv4-only Application Software |
| +----------------+    |
|                  |
```

Figure 2

A network-based Edge Translator performs the exact same as a host-based ET does, only that instead of assigning the IPv4 Service Addresses to an internal-only virtual network adapter, traffic destined for them are forwarded onto a network segment to which hosts that require IPv4 connectivity connect to. The ET also functions as the default IPv4 router for the hosts on this network segment.

Each host on the IPv4 network segment must acquire and assign an IPv4 Service Address to a local network interface. This document does not attempt to explore all the various methods by which this can be
accomplished, however one relatively straight-forward possibility would be to ensure the IPv4 Service Address(es) can be enclosed in an IPv4 prefix. The ET will then claim one address in this prefix for itself (used as the IPv4 default router address), and could assign the IPv4 Service Address(es) to the host(s) using DHCPv4. For example, if the IPv4 Service Addresses are 192.0.2.26 and 192.0.2.27, the ET would configure the address 192.0.2.25/29 on its IPv4-facing interface and would add the two IPv4 Service Addresses to its DHCPv4 pool.

One disadvantage of this method is that IPv4 communication between the IPv4 hosts and other services made available through SIIT-DC using the method described in Section 6 becomes impossible, if those other services are assigned IPv4 Service Addresses that also are covered by the same IPv4 prefix (e.g., 192.0.2.28). This is because the IPv4 nodes will mistakenly believe they have an on-link route to the entire prefix, and attempt to resolve the addresses using ARP (instead of forwarding them to the ET for translation to IPv6). This problem could however be overcome by avoiding assigning IPv4 Service Addresses which overlaps with an IPv4 prefix handled by an ET (at the expense of wasting some potential IPv4 Service Addresses), or by ensuring that they are only assigned to services which do not need to communicate with the IPv4 host(s) behind the ET.

Another way to avoid the problem is to use a private unrouted IPv4 network that does not encompass the IPv4 Service Addresses as the IPv4, and instead assign the IPv4 Service Addresses as secondary addresses on the servers. The ET must then route each IPv4 Service Address to its respective server using the server’s private on-link IPv4 address as the next-hop. This approach would ensure there are no overlaps, but on the other hand it would preclude the use of DHCPv4 for assigning the IPv4 Service Addresses, as well as create a need to ensure that the IPv4 application software is selecting the IPv4 Service Address (as opposed to its private on-link IPv4 address) as its source address when initiating outbound connections.

The basic ET illustrated in Figure 2 establishes an IPv4-only network segment behind itself. This is fine if the devices it provides IPv4 access have no support for IPv6 whatsoever; however if they are dual-stack capable, it is would not be ideal to take away their IPv6 connectivity. While it is recommended to use a host-based ET in this case, appropriate implementations of a host-based ET might not be available for every device. If the application protocol does not work correctly in a NAT environment, standard SIIT-DC cannot be used either. Thus, a network-based ET is the only solution.

The operator could avoid breaking the hosts’ IPv4 connectivity by connecting the ET’s IPv4 and IPv6 interfaces to the same network
segment, or by using a single dual-stacked interface instead. The latter alternative is shown in Figure 3. This could be thought of as an "ET on a stick". IPv6 traffic between the network and the hosts will bypass the ET entirely. IPv4 traffic from the hosts will be routed directly to the ET (because it’s their default IPv4 router), and translated to IPv6 before its being transmitted to the upstream default IPv6 router. The ET could attract inbound traffic to its IPv6 Service Addresses by responding to the upstream router’s IPv6 Neighbor Discovery [RFC4861] messages for them.

A Network-based Edge Translator "on a stick"

```
[IPv4 Internet]   [IPv6 Internet]
--+--<SIIT-DC GW>--+
 | [XLAT]
--+----------------+
[IPv6-only data centre network]
    ----<ET>------
      |   ____    |
      | /    \\   |
      | \____/   |
      |            |
      |            |
      +------------+

[Long-stack network segment]
++--<Dual-stack server>-----+
    +----------------+
    | AF_INET Dual-stack |
    | Application       |
    \--AF_INET6 Software +----------------+
    +----------------------------+
```

Figure 3
Yet another variation would be to implement the ET so that it transparently passes IPv6 traffic between its downstream and upstream network ports unmodified, e.g., using Layer-2 bridging. Packets sent to its own IPv6 Service Addresses from the upstream network are intercepted (e.g., by responding to IPv6 Neighbor Discovery [RFC4861] messages for them) and routed through the translation function, and forwarded out its downstream interface. The downstream network segment is thus becomes dual-stacked. This model is shown in Figure 4.

A Transparent Network-based Edge Translator

```
[IPv4 Internet]   [IPv6 Internet]
|                         |
+--|--<SIIT-DC GW>--+        |
    | [XLAT]            |
    +----------------+        |
    | [IPv6-only data centre network] |
    |                         |
    +--|--<Edge Translator>--+  |
        | _____________/      |
        | [Bridged IPv6] [XLAT] |
        /  /
        +----------------+
    [Dual-stack network segment]
    +--|--<Dual-stack server>----+
          |-------------------+   |
          +---AF_INET  Dual-stack  |
            | Application         |
            \--AF_INET6  Software  |
                  +----------------+
                  +----------------------------+
```

Figure 4

4. Detailed Topology Example

The following figure shows how an application (that is presumably incompatible with standard SIIT-DC) is being made available to the IPv4 Internet on the IPv4 address 192.0.2.4. The application will be able to know that this is its local address and thus be able to provide correct references to it in application payload.
The figure also shows how the same application is available over IPv6 on its IPv6 Service Address 2001:db8:12:34::3. This is included in order to illustrate how native IPv6 connectivity is not impacted by the Edge Translator, and also to illustrate how the address assigned to the ET (2001:db8:12:34::4) is separate from the primary IPv6 address of the server. It is however important to note that the application in question does not have to be dual-stack capable at all. IPv4-only applications would also be able to operate behind an ET in the exact same manner.

Note that the figure below could be considered a more detailed view of Customer A’s FTP server from the example topology figure in Appendix A of [I-D.ietf-v6ops-siit-dc]. Both figures intentionally use the exact same example IP addresses and prefixes.

SIIT-DC Host Architecture with Edge Translation
IPv6-capable user
+-------------------+         | IPv4-only user
| ============== |         | ============== |
+--------------<2001:db8::ab:cd>-+         +-<203.0.113.50>-+
|                                  |
| (the IPv6 internet)         (the IPv4 Internet)
|                                  |
|                                |
| +--------<192.0.2.0/24>-+         |
| |                                 |
| | SIIT-DC Gateway                 |
| | ==============                 |
| |                                |
| | Translation Prefix:            |
| | 2001:db8:46::/96             |
| |                                |
| | Static Address Mapping:        |
| | 192.0.2.4 <=> 2001:db8:12:34::4 |
| +--------<2001:db8:46::/96>-+         |
| |                                 |
| | (the IPv6-only data centre network) |
| |                                      |
| +--------<2001:db8:12:34::3>-<2001:db8:12:34::4>-++
| |                               |
| | IPv6-only server               |
| | ==============                 |
| |                                |
| | +--------<2001:db8:12:34::4>-+         |
| | |                                 |
| | | Edge Translator               |
| | | ==============                 |
| | |                                |
| | | Translation Prefix:           |
| | | 2001:db8:46::/96             |
| | |                                |
| | | Static Address Mapping:       |
| | | 192.0.2.4 <=> 2001:db8:12:34::4 |
| | +--------<192.0.2.4>-+         |
| | |                                 |
| | | [2001:db8:12:34::3]--------->[192.0.2.4]--
| | | AF_INET6     AF_INET           |
| | |                                |
| | | Dual-stacked application       |
| +----------------------------------------------+
5. Deployment Considerations

5.1. IPv6 Path MTU

The IPv6 Path MTU between the Edge Translator and the SIIT-DC Gateway will typically be larger than the default value defined in Section 4 of [RFC6145] (1280), as it will typically contain within a single administrative domain. Therefore, it is recommended that the IPv6 Path MTU configured in the ET is raised accordingly. It is RECOMMENDED that the ET and the SIIT-DC Gateway use identical configured IPv6 Path MTU values.

5.2. IPv4 MTU

In order to avoid IPv6 fragmentation, an Edge Translator should ensure that the IPv4 MTU used by applications or hosts is equal to the configured IPv6 Path MTU - 20, so that an maximum-sized IPv4 packet can fit in an unfragmented IPv6 packet. This ensures that the application may do its part in avoiding IP-level fragmentation from occurring, e.g., by segmenting/fragmenting outbound packets at the application layer, and advertising the maximum size its peer may use for inbound packets (e.g., through the use of the TCP MSS option).

A host-based ET could accomplish this by configuring this MTU value on the virtual network adapter, while a network-based ET could do so by advertising the MTU to its downstream hosts using the DHCPv4 Interface MTU Option [RFC2132].

5.3. IPv4 Identification Header

If the generation of IPv6 Atomic Fragments is disabled, the value of the IPv4 Identification header will be lost during the translation. Conversely, enabling the generation of IPv6 Atomic Fragments will ensure that the IPv4 Identification Header will carried end-to-end. Note that for this to work bi-directionally, IPv6 Atomic Fragment generation must be enabled on both the SIIT-DC Gateway(s) and on the Edge Translator.

Note that apart from certain diagnostic tools, there are few (if any) application protocols that make use of the IPv4 Identification header. Therefore, the loss of the IPv4 Identification value will therefore generally not cause any problems.

IPv6 Atomic Fragments and their impact on the IPv4 Identification header is further discussed in Section 4.8.2 of [I-D.ietf-v6ops-siit-dc].
6. Intra-DC IPv4 Communication

While SIIT-DC is primarily intended to facilitate communication between IPv4-only nodes on the Internet and services hosted in an IPv6-only network, it is also possible to facilitate communication between an IPv4-only service or application running behind an Edge Translator and another service/application made available over IPv4 through SIIT-DC. This other service/application may be a IPv6-only service, or it may also be an IPv4-only service running behind another ET.

Facilitating such communication requires that another Static Address Mapping is configured in the ET (one for each service it wants to communicate to). If there are two ETs involved, both of them must be configured in the same fashion for bi-directional communication to work. The following two subsections contain examples that demonstrate how this may be set up.

Note that for the intra-DC communication described in this section, the SIIT-DC Gateway is not involved at all. Therefore there is no requirement that the Static Address Mappings in question are also configured on the SIIT-DC Gateway. It is also possible to use private [RFC1918] IPv4 addresses, in order to reduce the need for publicly routable IPv4 addresses. However, if the IPv4-only application(s) are also to be made available to the IPv4 Internet through an SIIT-DC Gateway, it is highly recommended that the Static Address Mappings configured in the ET match those configured in the SIIT-DC Gateway. Otherwise one end up in the situation where a service is reached using different IPv4 addresses depending on whether one connects to it from the IPv4 Internet or from another IPv4-only application residing in the same data centre. While it may still work, the overall architecture gets significantly more complex.

Finally, if both services/applications support IPv6, it is highly recommended that IPv6 is used for all internal communications. The approach described in this section should only be used if one or both of the services or applications only supports IPv4, making native IPv6 communication impossible.

6.1. Between IPv4-Only and IPv6-Only Services

This section demonstrates how an IPv4-only service/application "A" running behind an ET can communicate with an IPv6-only service "B".

Intra-DC IPv4-only to IPv6-only Overview
In this example, the IPv4-only application on server "A" is listening on the IPv4 address 192.0.2.6, which is made available to the IPv6 network on the IPv6 address 2001:db8:6:: (by the ET). The IPv6-only application on server "B" is only listening on the IPv6 address 2001:db8:7::, and has no knowledge of IPv4.

In order to facilitate communication between the two application, another Static Address Mapping must be configured in the ET on server "A". This provides an IPv4 address (192.0.2.7) that the IPv4-only application can communicate with, which represents the IPv6 address used by application "B" (2001:db8:7::).

The following figure shows the packet translations step by step, for a packet sent by the IPv4-only application "A" to the IPv6-only application "B". For traffic in the opposite direction, you may read the figure from the bottom up and swap the Src/Dst addresses.
6.2. Between Two IPv4-Only Services

This section demonstrates how an IPv4-only service/application "A" running behind an ET can communicate with an IPv4-only service/application "B" running behind another ET.

Intra-DC IPv4-only to IPv6-only Overview
In this example, the IPv4-only application on server "A" is listening on the IPv4 address 192.0.2.8, which is made available to the IPv6 network on the IPv6 address 2001:db8:8:: (by the ET). In the same fashion, the IPv4-only application on server "B" is listening on the IPv4 address 192.0.2.9 and is made available by its ET on the IPv6 address 2001:db8:9::.

In order to facilitate communication between the two applications, a second Static Address Mapping must be configured in the ET on both servers. This provides each application with an IPv4 address that represents the other application. Thus bi-directional communication between the two applications can commence.

The following figure shows the packet translations step by step, for a packet sent by the IPv4-only application "A" to the IPv4-only application "B". For traffic in the opposite direction, you may read the figure from the bottom up and swap the Src/Dst addresses.
7. Acknowledgements

The author would like to especially thank the authors of 464XLAT [RFC6877]: Masataka Mawatari, Masanobu Kawashima, and Cameron Byrne. The architecture described by this document is merely an adaptation of their work to a data centre environment, and could not have happened without them.

The author would like also to thank the following individuals for their contributions, suggestions, corrections, and criticisms: Fred Baker, Tobias Brox, Ray Hunter, Shucheng LIU (Will), Andrew Yourtchenko.

8. IANA Considerations

This draft makes no request of the IANA. The RFC Editor may remove this section prior to publication.
9. Security Considerations

This section discusses security considerations specific to the use of an Edge Translator. See the Security Considerations section in [I-D.ietf-v6ops-siit-dc] for additional security considerations applicable to the SIIT-DC architecture in general.

9.1. Address Spoofing

If the ET receives an IPv4 packet from the application from a different source address than the one it has a Static Address Mapping for, the both the source and destination addresses will be rewritten according to [RFC6052]. After undergoing the reverse translation in the SIIT-DC Gateway, the resulting IPv4 packet routed to the IPv4 network will have a spoofed IPv4 source address. The ET should therefore ensure that ingress filtering (cf. BCP38 [RFC2827]) is used on the ET’s IPv4 interface, so that such packets are immediately discarded.

If the ET receives an IPv6 packet with both the source and destination address equal to the one it has a Static Address Mapping for, the resulting packet would appear to the application as locally generated, as both the source address and the destination address will be the same address as the one configured on the virtual IPv4 interface. This could trick the application into thinking this packet came from a trusted source, and give elevated privileges accordingly. To prevent this, the ET should discard any received IPv6 packets that have a source address that is equal either to either the IPv4 (after undergoing [RFC6052] translation) or the IPv6 address in the Static Address Mapping.

10. References

10.1. Normative References

[I-D.ietf-v6ops-siit-dc]


10.2. Informative References


Author’s Address

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
Abstract

The IPv6 address range of 0::/64 is reserved for loopback addresses. This expands from the single loopback address already defined for IPv6, ::1, to allow for a set of addresses to be used when packets are intended to stay within a host system. Multiple loopback addresses allow for simultaneous varied uses of the loopback addresses as has proven, albeit in limited ways, in IPv4. And exception is made to accommodate the ::0/128, already defined as The Unspecified Address.

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 1, 2015

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

0.  NOTE TO RFC EDITOR AND REVIEWERS ........................................ 1
1.  Introduction ............................................................ 1
2.  Use of ::0/64 Addresses ................................................. 2
3.  IANA Considerations .................................................... 1
4.  Security Considerations ................................................ 1
5.  Acknowledgements ...................................................... 1
6.  References ............................................................. 1
   6.1.  Normative References .............................................. 1
Authors’ Addresses ......................................................... 1

0.  NOTE TO RFC EDITOR AND REVIEWERS

This section should be removed prior to publication.

1.  Introduction
The "IP Version 6 Addressing Architecture" [RFC 4291] defines a single IPv6 loopback address as ::1/128. In "Special-Purpose IP Address Registries" [RFC6890], 127.0.0.0/8 is assigned for loopback addresses, with usually just 127.0.0.1/32 implemented by default.

Ordinarily, just one address (whether IPv4 or IPv6) is sufficient for loopback addressing on a node but there have been a few use cases showing that it is desirable to have more than 1 (but less than the over 16 million that are in an IPv4 /8).

One use case is testing or prototyping, desiring to mimic a small network of processes on one node. To demonstrate a particular protocol's server running on a well-known port, having multiple addresses where packets can "travel" within the host is useful.

Another use case has arisen from ICANN's Controlled Interruption approach [need reference] which directs errant traffic to a loopback address with two distinct goals in mind. One is to prevent the leakage of packets that are known to be erroneously sent and two is to leave "bread crumbs" in log files for operators to use to help track why the erroneous packets are being sent.

The use of ::0/64 is (proposed) to represent an address range (or block) encompassing The Unspecified Address and loopback addresses.

2. Use of ::0/64 Addresses

The Unspecified Address, or ::0/128, remains as defined in RFC 4291’s section 2.5.2. That definition is included by reference here so as to prevent any unintentional changes to the original text.

For all other addresses within ::0/64, the rules for using are the same as the rules in RFC 4291’s section 2.5.3, again included by reference so as not to introduce any unintentional changes.

3. IANA Considerations

Registration in the IANA IPv6 Special-Purpose Address Registry

The IANA is directed to add ::0/64 to the "IANA IPv6 Special-Purpose Address Registry" specified in [RFC6890] as follows:

Address Block: ::0/64

Name: Loopback and Unspecified Addresses

RFC: [THIS DOCUMENT]

Allocation Date: [APPROVAL DATE]

Termination Date: N/A

Source: True [1]

Destination: False

Forwardable: False

Global: False

Reserved-by-Protocol: True

[1] True for ::0/128, False for all other addresses in ::0/64

The IANA is directed to remove Table 17 and Table 18 as defined in RFC 6890, section 2.2.3.
4. Security Considerations

Security is not (yet) a consideration

5. Acknowledgements

We all this all to David Conrad.

6. References

6.1. Normative References

[RFC 6890] "Special-Purpose IP Address Registries:, Cotton, Vegoda, Bonica & Haberman, Apr 2013

Authors’ Addresses

Edward Lewis
edward.lewis@icann.org
801 17th Street NW
Suite 400
Washington, DC, 20006
US

Lewis Expires September 1, 2015
Multiple IPv6 Prefixes: Background and Considerations

draft-liu-v6ops-running-multiple-prefixes-03

Abstract

This document describes several typical multiple prefixes use cases, and discusses that running multiple IPv6 prefixes/addresses in one network/host should be common practice that administrators need to adapt.

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 26, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

In IPv6 networks, there are deployment scenarios in which multiple prefixes coexist simultaneously in one network. Several typical use cases are:

- Multiple Prefixes with Different Scopes (described in Section 2.1)
- IPv6 multihoming based on multiple PA prefixes (described in Section 2.2)
- Make-before-break renumbering (described in Section 2.3)
- An IPv6 network with multiple services, each of which has a distinct prefix (described in Section 2.4).

To support the multiple prefixes running mode, there have been some technologies developed. This document discusses these technologies of different aspects, which could allow and smoothen the multiple prefix operation.

Note that, although MIF (Multiple InterFaces) [RFC6418] architecture also involves multiple IPv6 prefixes, it mainly targets different interfaces which attach to different networks respectively. This document discusses the multiple IPv6 prefixes running in the same network.
2. Multiple Prefixes Use cases

2.1. Multiple Prefixes with Different Scopes

IPv6 contains link-local addresses, global addresses and unique local addresses, which by definition are global but normally are site-scope by practice.

As specified in [RFC4291], all interfaces are required to have at least one Link-Local unicast address. This is the basic case of running multiple prefixes. However, this does not require operations from the network administrators since it is automatically processed.

Besides Link-Local addresses, the Unique Local Addresses (ULAs, [RFC4193]) might also be used for the internal communication within a site network. In many deployment, the ULA is used along with PA (Provider Aggregated) addresses, which connect to the public network. The benefit of such combination is to provide separate local communication from the globally communication so that the local communication would not be impacted when ISP uplink fail or prefix(es) be renumbered. It is especially beneficial for the home network and private OAM plane or internal-only nodes in an enterprise.

2.2. Multihoming based on Multiple PA Prefixes

When a network is multihomed, the multiple upstream network providers would assign prefixes respectively. If a network does not acquire a PI (Provider Independent) address space, multihoming will result coexistent multiple PA prefixes. In such network, a single host have multiple PA IPv6 addresses that associated with different prefixes.

This scenario rarely exists in IPv4 networks, since IPv4 only allows single address per interface. But it is quite practical in IPv6. This new feature of IPv6 allows the SMEs (Small/Medium Enterprises) to multihome without the burden of running PI address space or running IPv6 NAT. Furthermore, multiple PA spaces do not have the potential global routing system scalable issue as the PI does [RFC4894].

However, multihoming with multiple PA prefixes has some operational issues which mainly include address selection, next-hop selection, and exit-router selection. For detailed discussion, please refer to [RFC7157]. [Editor’s note: more discussion to be filled.]
2.3. Multiple Prefix Co-existing during Network Renumbering

[RFC4192] describes a procedure that can be used to renumber a network from one prefix to another smoothly through a "make-before-break" transition. In the transition period, both the old and new prefixes are available; the usage of multiple prefixes provides the smooth transition and avoids the session outage issue in most of renumbering operations.

2.4. Service Prefixes

An IPv6 network may simultaneously provide multiple services, such as IPTV, Internet access, VPN, etc. Each of these services should have a distinct prefix. The network may apply different policy based on the distinguished prefixes. This deployment would simplify the management and processing on network devices, such as forwarding routers, access authentication devices, account devices, border filter, etc. The ISPs would provide one subscriber multiple addresses/prefixes to access different services. This deployment would particularly benefit for traffic recognition and management.

3. Operational Availability and Considerations

This section discusses some technologies of different aspects, which could allow and smooth the multiple prefix operation.

3.1. Multiple prefix provisioning

- Multiple Prefixes from Different Provisioning Domains

In [I-D.ietf-mif-mpvd-arch], provisioning domain is defined as consistent set of network configuration information. Classically, the entire set available on a single interface is provided by a single source, such as network administrator, and can therefore be treated as a single provisioning domain.

But in modern IPv6 networks, multihoming or service prefixes may result in provisioning information from more than one provisioning domains being presented on a single link. In these scenarios, current technologies lack support of distinguishing information from multiple provisioning domains, thus the host would not be able to associate configuration information with provisioning domains.

However, there are several techniques under developing in MIF WG to solve the problems, we could expect them to be standardized in the near future.
Co-existing DHCPv6/SLAAC

Both SLAAC [RFC4862] and DHCPv6-PD [RFC3633] could assign IPv6 prefixes. DHCPv6-PD is normally run between routers and routers or routers and DHCPv6 [RFC3315] servers; while SLAAC is normally run between routers and downstream hosts. The two protocols could collaborate sufficiently to cover the whole network's prefix provisioning.

If operated properly, SLAAC and DHCPv6 could also co-exist for IPv6 addresses provisioning based on different prefixes. They need to carefully deal with the interaction between the two protocols. It is mostly regarding to the M flag in Neighbor Discovery [RFC4861] messages.

3.2. Address Selection

In order to support multiple addresses well, IPv6 introduced address selection mechanism which utilizes a address selection policy table to calculate a proper source address for a given destination address. Of course, destination addresses selection is also defined. [RFC6724] described the rationale and algorithms in detail, and also defined a default address selection policy table for operating systems.

Note that, the [RFC6724] is a replacement of the old [RFC3484] specification to improve some behaviors (e.g. to prefer IPv4 over ULA for outside connectivity). Currently, so far there haven't been many operating systems supporting the new standard, but we could expect that the new standard would be available in all new released operating systems and becomes the mainstream in the near future.

3.3. Exit-router selection

In multiple PA multihoming networks, if the ISPs enable ingress filtering at the edge (BCP38, [RFC2827]), then there comes the exit router selection issues that outgoing packets are routed to the appropriate border router and ISP link. Normally, a packet sourced from an address assigned by ISP X should not be sent via ISP Y, otherwise it would be filtered by ISP Y.

In the past, the administrators have to either communicate with the ISP for not filtering the prefixes or manually configure routing policies within the network to make sure the traffics are forwarded to the right upstream link, based on source prefixes. Now, there are some source-based routing technologies under development and standardization. We could expect these solutions available soon.
4. Security Considerations

This document does not introduce any new mechanisms or protocols technologies and as such does not introduce any new security threads. Nevertheless, relevant important security considerations are worth to be iterated here:

- [RFC7157] gives the security considerations for multi-prefix based multihoming.
- Address selection relevant security considerations are described in [RFC6724].
- ND cache exhaustion caused by multiple addresses per host in a big L2 network is described in Section 3.2. It is possibility that malicious users intentionally configure massive addresses on host to make the gateway ND cache exhausted. So administrators always need to consider mitigation operations for potential ND cache DoS attack which is documented as [RFC6583].

5. IANA Considerations

This draft does not request any IANA action.

6. Acknowledgements

Valuable inputs of the texts/ideas were from Ole Troan.

Useful comments were received from Brian Carpenter, Victor Kuarsingh, Lorenzo Colliti, Mikael Abrahamsson, Fred Baker, Lee Howard and Roberta Maglione.

This document was produced using the xml2rfc tool [RFC2629]. (initially prepared using 2-Word-v2.0.template.dot. )

7. References

7.1. Normative References


7.2. Informative References

[I-D.ietf-mif-mpvd-arch]
Anipko, D., "Multiple Provisioning Domain Architecture",
draft-ietf-mif-mpvd-arch-11 (work in progress), March 2015.


Authors’ Addresses

Bing Liu
Huawei Technologies
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: leo.liubing@huawei.com

Sheng Jiang
Huawei Technologies
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: jiangsheng@huawei.com

Bo Yang
Huawei Technologies
Q21, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: boyang.bo@huawei.com
HTTP State Management Mechanisms with Multiple Addresses User Agents

draft-vyncke-v6ops-happy-eyeballs-cookie-01

Abstract

HTTP servers usually save session states in their persistent storage indexed by session cookies generated by the HTTP servers. It is up to the HTTP user-agent to send this session cookie on each HTTP request. Some HTTP servers check whether the cookie is associated with the HTTP user-agent by the means of the user-agent IP address. Everything linking a state to an IP address (such as OAuth access code) to an IP address has the same issue.

If the Happy Eyeball mechanism is used to select between IPv6 and IPv4, it may happen that while using the same HTTP server, some HTTP requests are done over IPv6 and the others over IPv4, which leads to two different sets of session states in the HTTP server. This has the consequence of inconsistencies at the HTTP server.

The only purpose of this document is to document this issue in more details than in section 8.2 of RFC 6883 including security considerations and mitigations.

A similar problem arises with the use of non RFC 6888 compliant Carrier-Grade NAT (CGN) devices used to access an IPv4-only HTTP server or HTTP user-agent using multi-homing.

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."
HTTP requests are basically stateless, therefore if a HTTP server requires to have some states associated to a HTTP user-agent (such as user name, login state, history, shopping basket, ...), there is a need to conserve those states. This is usually done by using a HTTP cookie (see also RFC6265 [RFC6265]) identifying the session; also called "session state cookie".

This session state cookie is generated by the HTTP server at the very first HTTP request from a HTTP user-agent. The cookie is usually opaque (often a random number) and has no semantic except as being an index within the persistent storage of the HTTP server. This index is used to access the complete state of the user-agent. This mechanism is secure if the cookie is transferred with confidentiality
between the server and the user-agent. If the cookie transfer and storage are not secured, then any hostile user-agent can reuse this cookie to access the full original session states (including shopping basket, payment details, ...); this attack is called ‘session cookie stealing’. This attack can happen if the HTTP traffic is intercepted by a man-in-the-middle attack but a good use of Transport Level Security RFC5246 [RFC5246] can prevent it. The attack can also happen with some hostile scripting or other pieces of malware running on the user agent, that could copy and send the session cookie to the hostile user-agent; hence, it is not enough to use TLS to secure the session cookies.

Some HTTP applications link the user-agent IP address (whether IPv6 or IPv4) to the session state, probably for additional security checks in order to prevent session cookie stealing. This link leads to some issues in a dual-stack world which are described in this document.

The author knows about at least two large web sites having this problem. It was so severe that those sites which were dual-stack had to move back to being IPv4-only... until the application and its security is updated.

1.1. Other Use of Session Cookies

Beside the use of session cookies by the HTTP server to keep states on the server, the very same cookie is also sometimes used by Server Load Balancing (SLB) mechanism to ensure that all HTTP requests from the same user-agent (even if behind a NAT) are always sent to the same physical HTTP server. This is required if the server persistent storage is local to the server and is not shared by all the physical servers behind the SLB.

1.2. new section

Actually the problem is more generic than the session cookie, everything linking a state to an IP address has the same issue. This includes OAuth [RFC6749] access tokens, bearer tokens, ... but also other mechanisms such as rate limiting per IP address or access control per IP address (for instance a captive portal for a guest net).

2. Issues

Similar issues can be caused by Happy Eyeball RFC6555 [RFC6555], Carrier-Grade NAT (CGN) and having multiple interface or being multi-homed.
2.1. Happy Eyeballs Issue

When a HTTP user-agent uses the Happy Eyeball mechanism to access a HTTP server, then, part of the HTTP requests can happen over IPv6 and another part over IPv4 if the latency between IPv4 and IPv6 varies quickly over time. If there is a link between the session cookie and the user-agent IP address, then upon the first change of IP protocol version, the states associated to the cookie will be invalidated and will be deleted. Here is an example:

1. User-agent with IPv4 address, ADDR4, connect to the server by using IPv4 because IPv6 is slower; the first request does not have any HTTP cookie;

2. Server generates a new cookie C4 and stores in its persistent storage that C4 is associated with address ADDR4;

3. User-agent continues his/her session using IPv4, on each new request the HTTP server receives the cookie C4 and checks that the user-agent address is indeed ADDR4;

4. Latency of IPv6 changes and becomes now faster than IPv4;

5. User-agent now uses its IPv6 address, ADDR6, to connect to the same server and continues to use the same cookie C4 as the server name is unchanged;

6. The server receives the HTTP request with the C4 cookie and checks whether C4 is associated with ADDR6 which is not the case... All session states are deleted and a new cookie, C6, is generated and associated to the IPv6 address ADDR6;

7. The end-user becomes frustrated because he/she has to restart his/her complete session from the beginning.

This cookie invalidation may have some security benefit but it actually prevents a host using Happy Eyeballs to have a persistent session with a dual-stack HTTP server; with painful consequences for the user-experience: disconnection, loss of shopping basket, ...

2.2. Carrier-Grade NAT Issue

RFC6888 [RFC6888] describes the CGN requirements but not all CGN implement them. Some CGN in the real world have a pool of IPv4 addresses and do not always use the same public IPv4 address for all requests from a CGN client. This obviously leads to the same problem as in section Section 2.1. This will happen for IPv4-only HTTP servers.
Whether the CGN is used by IPv4 clients or by IPv6 clients (via NAT64 RFC6146 [RFC6146]) does not make any difference to the problem. The use of the address family translation by MAP-T MAP-T [I-D.ietf-softwire-map-t] does not suffer from this issue for IPv4-only HTTP servers since one subscriber is restricted to several layer-4 ports from a single IPv4 address.

2.3. Multiple Interfaces Issue

When the HTTP user-agent has multiple interfaces, for example 3GPP and Wi-Fi, the preferred IP address depends on the WiFi or 3GPP availability. In this case, a similar issue to Section 2.1 also happens as the session cookie can be linked first to the Wi-Fi IP address then when the user-agent looses its Wi-Fi connectivity the session cookie will be overwritten by a new session cookie linked to the 3GPP address.

Whether the user-agent uses IPv4-only, IPv6-only or dual-stack has no impact on the issue.

3. Mitigations

The obvious mitigation for this issue is NOT to link any HTTP state management (including cookies) to any IP address of the HTTP user-agent at the risk of increasing the risk of "session cookie stealing".

The author also believes that:

Multipath TCP RFC6824 [RFC6824] hides completely the set of addresses of the client to the application. Only the first subflow’s IP addresses are exposed to the application, even if a later subflow uses a different address family; so, any session cookie will be permanently linked to the first IP address used by the HTTP user-agent;

HTTP/2 [I-D.ietf-httpbis-http2] multiplexes multiple HTTP sessions over a single TCP connection, therefore, Happy Eyeball (or bad CGN) sees only one TCP connection and a change of IP address will never occur during the lifetime of this TCP connection.

4. IANA Considerations

This document contains no IANA considerations.
5. Security Considerations

The association of the session cookie with the user-agent IP address has some security value as it can help prevent "session cookie stealing" in some limited situations; this benefit should be balanced with the lack of persistent session and the remaining vulnerability if the HTTP session can be intercepted by a man-in-the-middle attack. Moreover with more and more CGN being deployed, linked a session cookie to an IP address shared by hundreds of subscribers is less effective as the cookie could be reused by any subscribers using the same shared public IP address.

6. Acknowledgements

The author would like to thank Brian Carpenter, Ray Hunter, Jeroen Massar, Dan Metzler, Erik Nygren, Mark ZZZ Smith, Joe Touch, Dan Wing and Andrew Yourtchenko for some discussions on this topic. Of course, RFC6883 [RFC6883] has already mentioned this issue without many details.

7. Informative References

[I-D.ietf-httpbis-http2]

[I-D.ietf-softwire-map-t]


Internet-Draft HTTP States with Multiple Addresses March 2015


Author’s Address

Eric Vyncke
Cisco
De Kleetlaan 6a
Diegem 1831
Belgium

Phone: +32 2 778 4677
Email: evyncke@cisco.com