Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 13, 2013

T. Anderson
Redpill Linpro

November 9, 2012

Stateless IP/ICMP Translation in IPv6 Data Centre Environments
draft-anderson-siit-dc-00

Abstract

This document describes the use of Stateless IP/ICMP Translation (SIIT) in data centre environments in order to simultaneously facilitate IPv6 deployment and IPv4 address conservation. It describes the overall architecture, and provides guidelines for both operators and implementers.

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 May 13, 2013.

Copyright Notice

Table of Contents

1. Introduction

This document describes deploying SIIT [RFC6145] as a network-centric stateless translation service that allow a data centre operator or Internet content provider run his 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] has been the recommended way to transition from an IPv4-only environment to one capable of serving IPv6 users. For data centre and Internet content providers, however, dual stack operation has a number of disadvantages compared to single stack operation, in particular increased complexity and operational overhead, and very low expected return of investment in the short to medium term, as there are practically no end users who have only 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, a better approach was needed. The design goals were, in no particular order:

- To promote the deployment of native IPv6 services
- To 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

SIIT meets all of these requirements, which will be elaborated on in the following subsections.

1.1.1. Single Stack IPv6 Operation

SIIT 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 the provider's services and applications.

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., NAT44 [RFC3022]), a SIIT gateway does not keep any state between each packet in a single connection/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 gateways to easily attain "line rate" performance, it also allows for per-packet load balancing between multiple gateways using Equal-Cost Multipath Routing [RFC2991]. Asymmetric routing is also unproblematic, which makes it easy to avoid traffic trampolines, as the prefixes involved may be anycasted from all the SIIT gateways in the provider's network.

Finally, stateless operation means that high availability is easily achieved. If an SIIT gateway should fail, its traffic can be re-routed onto another SIIT gateway using completely standard IP routing protocols. This will not impact existing flows any more than what any other IP re-routing event would.

1.1.3. No Loss of End User's Source Address

SIIT will map the entire end-user's source address into an predefined IPv6 translation prefix. This allows the application server to identify the user by his IPv4 address, which is useful for performing tasks like Geo-Location, logging, abuse handling, and so forth.

1.1.4. No Forklift Upgrades Required

Except for the introduction of the SIIT gateways themselves, there is no change required in the network, servers, applications, or anywhere else to specifically support SIIT, compared to a dual stack deployment. From the clients', the servers', the IPv6 data centre network's, and the IPv4 Internet's point of view, SIIT is practically invisible. It will work with any standards-compliant IPv4 or IPv6 stack.

1.1.5. No Architectural Dependency on IPv4

SIIT will allow an ICP or data centre operator to build their 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 gateways. Therefore, by deploying native IPv6 along with SIIT, operators will avoid future migration or deployment projects relating to IPv6 roll-out and/or IPv4 sun-setting.

1.2. Comparison to Other IPv6 Migration Strategies

1.2.1. IPv4-only Service with Translation for IPv6 Users

Typically, this migration strategy involves having an IPv4-only application stack, with some device in front that the IPv6 client connect to, who will then translate or proxy the traffic to the IPv4-only system. This approach is probably the easiest to retrofit to an existing IPv6 service environment, however it does have a few shortcomings not shared by SIIT. In particular:

- No conservation of IPv4 addresses

- The translator/proxy must be a stateful device, requiring traffic to flow symmetrically across a single instance, in turn giving the solution poor scaling properties and routing flexibility

- A fail-over event will disrupt all active flows, unless there is some state replication mechanism (which would likely increase complexity and hurt performance and scaling properties)

- Loss of the client's source IP address, if it cannot be injected into application-layer headers such as HTTP's X-Forwarded-For (which is impossible if the application layer is using encryption)

1.2.2. Dual Stack

Dual stack, unlike SIIT, considerably increases complexity and operational overhead compared to single stack operation for a number of reasons. Some examples of this include:

- Duplicate work for design, set-up, documentation, and monitoring

- Duplicate ACLs in both network components and applications

- An exponential increase in possible failure scenarios

- Increased application development and maintenance costs

- Increased need for staff training and competency

Furthermore, dual stack does not help conserve IPv4 addresses.

2. Architectural Overview

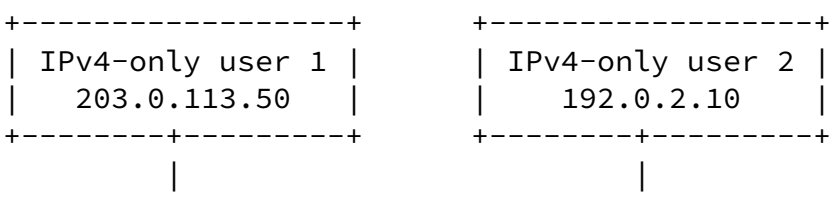
This section attempts to explain the basic SIIT architecture by describing an example topology of a data centre hosting two IPv6-only customers:

- Alice, operating a publicly available web service.

o Bob, operating publicly available DNS and MX service.

Since both Alice and Bob's server installations contain other servers that provide internal services, if they had used IPv4, they each would have needed their server LANs to be provisioned with at minimum a /29, thereby consuming 16 IPv4 addresses. With SIIT, the IPv4 address consumption is reduced to 3 - the same number of publicly available services.

Example SIIT Topology



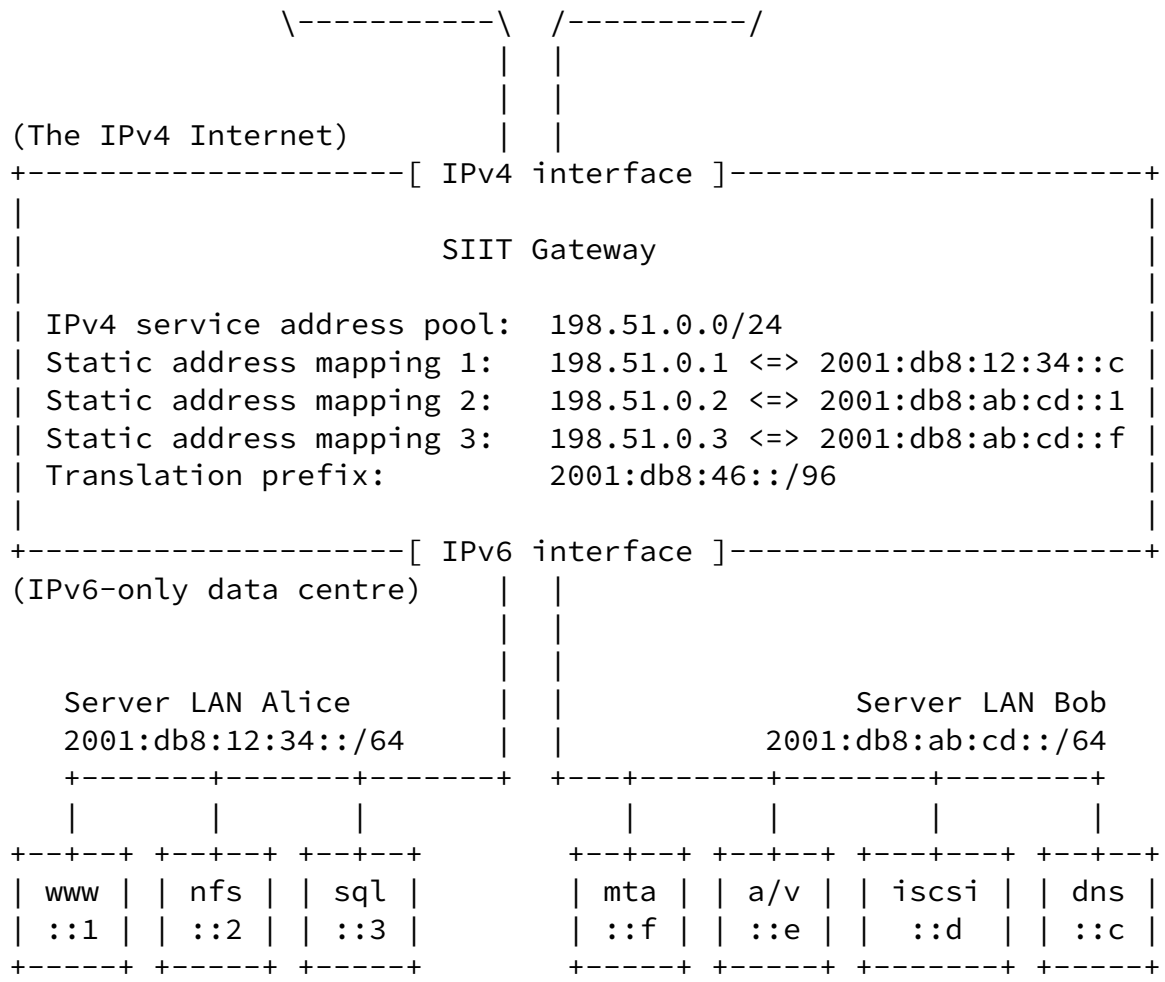


Figure 1

198.51.0.0/24 is allocated as a pool from which individual IPv4 service addresses are drawn. The provider must route this prefix to the SIIT 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 gateway's IPv4 interface.

The static address mappings are used for translating the service's IPv6 address into IPv4 and vice versa. When translating from IPv4 to IPv6, any IPv4 address found in the list of static mappings will be

rewritten to its corresponding IPv6 address, and vice versa when

translating from IPv6 to IPv4.

2001:db8:46::/96 is the IPv6 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]. This algorithmic mapping has a lower precedence than the static mappings.

The SIIT 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 gateways may exist simultaneously in an operators infrastructure, as long as they all have the same translation prefix and list of static mappings configured.

2.1. DNS Configuration

The native IPv6 address of the publicly available services should be registered in DNS using AAAA records, while the corresponding IPv4 address (according to the static mapping), should be registered using an A record. This results in the following DNS records:

```
www.alice.tld.    IN AAAA  2001:db8:12:34::1
www.alice.tld.    IN A     198.51.0.2

mta.bob.tld.      IN AAAA  2001:db8:ab:cd::f
mta.bob.tld.      IN A     198.51.0.3

dns.bob.tld.      IN AAAA  2001:db8:12:34::c
dns.bob.tld.      IN A     198.51.0.1
```

2.2. Example Packet Flow

In this example, "IPv4-only user 2" initiates a request to Alice's web server. He starts by looking up the IPv4 address of "www.alice.tld" in DNS, and attempts to connect to this address on port 80 by transmitting the following IPv4 packet:

```
+-----------------------------------------------+
| IP Version:           4                       |
| Source Address:       192.0.2.10              |
| Destination Address:  198.51.0.2              |
| Protocol:             TCP                     |
|-----------------------------------------------|
| TCP SYN [...]                                 |
+-----------------------------------------------+
```

This packet is then routed over the Internet to the (nearest) SIIT

gateway, which will translate it into the following IPv6 packet and forward it into the IPv6 network:

```
+-----------------------------------------------+
| IP Version:           6                       |
| Source Address:       2001:db8:46::192.0.2.10 |
| Destination Address:  2001:db8:12:34::1       |
| Next Header:          TCP                     |
|-----------------------------------------------|
| TCP SYN [...]                                 |
+-----------------------------------------------+
```

The destination address was translated according to the configured static mapping, while the source address was translated according to the [RFC6052] mapping (because it did not match any static mappings). 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::192.0.2.10 may also be expressed as 2001:db8:46::c000:20a, cf. section 2.2 of [RFC2373].

Next, Alice's web server receives this IPv6 packet and responds to it like it would with any other IPv6 packet:

```
+-----------------------------------------------+
| IP Version:           6                       |
| Source Address:       2001:db8:12:34::1       |
| Destination Address:  2001:db8:46::192.0.2.10 |
| Next Header:          TCP                     |
|-----------------------------------------------|
| TCP SYN+ACK [...]                             |
+-----------------------------------------------+
```

The response packet is routed to the (nearest) SIIT gateway's IPv6 interface, which will translate it back to IPv4 as follows:

```
+-----------------------------------------------+
| IP Version:           4                       |
| Source Address:       198.51.0.2              |
| Destination Address:  192.0.2.10              |
| Protocol:             TCP                     |
|-----------------------------------------------|
| TCP SYN+ACK [...]                             |
+-----------------------------------------------+
```

This time, the source address matched the static mapping, while the destination address was translated according to [RFC6052]. 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.

3. Deployment Guidelines for Operators

3.1. Choice of Application

As noted in [RFC2663], [RFC2993], and [RFC3022], higher-level protocols that embed addresses as part of their payload, will most likely not work through any form of address translation, including SIIT. As a general rule, if an application layer protocol does work through standard NAT44 (see [RFC3235]), it will most likely work through SIIT as well.

It is recommended that an initial deployment of SIIT is used for applications where IPv4-only nodes on the Internet initiate traffic towards the IPv6-only services. While it is possible to combine SIIT with DNS64 [RFC6147] or similar mechanisms in order to allow an IPv6-only server to initiate communication with IPv4 nodes through an SIIT gateway, this may be more complicated to implement, as the server must ensure to always use the address statically mapped on the SIIT gateway as the source when initiating communication.

In particular, HTTP [RFC2616] is a good choice of an application protocol to start deploying SIIT with, as it is both ubiquitous and known to work very well through address translation.

Note that implementations of SIIT may bundle Application Level Gateways (ALGs) to add specific support for certain application protocols that would otherwise break, similar to what is commonly done with NAT44 implementations. If ALGs are being used, care must be taken to ensure that all the translators in the network all have compatible ALGs.

3.2. 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 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 sell SIIT service to other operators.

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 prefix may use any of the lengths described in section 2.2 of [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::192.0.2.10" (see section 2.4 of [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.

3.3. Routing Considerations

The IPv4 service address prefix(es) and the IPv6 translation prefix

may be routed to the SIIT gateway(s) as any other IPv4 or IPv6 route in the provider's network.

If more than one SIIT 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 translator, reducing or eliminating traffic trampolines, as well as provide high availability - if one translator fails, the dynamic routing protocol will automatically redirect the traffic to the next-best translator.

3.4. Location of the Translators

In order to prevent traffic trampolines, it is optimal to place the translators as close as possible to the direct path between the

servers and the end users.

Ideally, they are implemented as a logical function within the IP routers would handle the traffic anyway (if it wasn't to be translated). This way, the translation service would not need separate networks ports to be assigned (which might become saturated and impacted the service), nor would it need extra rack space or energy. Some good choices of the location could be within a data centre's access routers, or inside the provider's border routers. If every single application in the data centre or the provider's network eventually get single-stacked, there would no need to run IPv4 on the inside of the translators - thus allowing the operator to reclaim IPv4 addresses from the network infrastructure that may instead be used for translated services.

3.5. 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 works equally well for a dual stacked servers, which makes migration easy - after setting up the translation function, the DNS 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.

For high-volume services migrating to SIIT from dual stack, DNS Round Robin may be used to gradually migrate the service's IPv4 traffic from its native IPv4 address(es) to the translated one(s).

3.6. Packet Size and Fragmentation Considerations

There are two key differences between IPv4 and IPv6 relating to packet sizes that one should consider when deploying SIIT. They result in a few problematic corner cases, which can be dealt with in a few different ways.

The operator may find that relying on fragmentation in the IPv6 domain is undesired or even operationally impossible [FRAGDROP]. For this reason, the recommendations in this section seeks to minimise the use of IPv6 fragmentation.

Unless otherwise stated, this section assumes that the MTU in both the IPv4 and IPv6 domains is 1500 bytes.

3.6.1. IP 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 gateway. The only way to avoid this is to increase the Path MTU between the SIIT 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. However, if the server's IPv6 implementation allows the MTU to be set differently for specific destinations, it MAY be increased to 1520 for destinations within the translation prefix specifically.

3.6.2. Minimum Path MTU Difference

The minimum allowed MTU in IPv6 is 1280 bytes, while no such restriction exists in IPv4. This means that an 1280 byte large IPv6 packet sent to an IPv4 client may need to be fragmented by a router in the IPv4 network.

By default, an SIIT gateway will set the DF flag when translating from IPv6 to IPv4, resulting in a situation where the IPv6 server may receive an ICMPv6 Packet Too Big where the indicated MTU value is less than the IPv6 minimum of 1280. In this situation, the IPv6 server has two choices on how to proceed, according to the last paragraph of section 5 of [RFC2460]:

- It may reduce its Path MTU value to the value indicated in the Packet Too Big. This causes no problems for the SIIT function.

- It may reduce its Path MTU value to 1280, and also include a Fragmentation header in each subsequent packet sent to that destination. This instructs the SIIT gateway to clear the DF flag in the resulting IPv4 packet, and also provides the Identification value.

If the use of the IPv6 Fragmentation header is problematic, and the operator has IPv6 servers that implement the second option above, the operator should enable a feature on the SIIT gateways which ensures that the resulting MTU field is always set to 1280 or higher when translating ICMPv4 Need to Fragment into ICMPv6 Packet Too Big, and that when translating IPv6 packets smaller or equal to 1280 bytes the resulting IPv4 packets will have the DF flag cleared and an Identification value generated, cf. Section 4.5.

3.6.3. "Atomic Fragments"

By default, an SIIT gateway will include a Fragmentation header in the resulting IPv6 packet when translating from an IPv4 packet with the DF flag cleared, cf. section 4 of [RFC6145].

This happens even though the resulting IPv6 packets aren't actually fragmented into several pieces, resulting in "Atomic Fragments" [ATOMFRAG]. This is generally not useful in a data centre environment, and it is therefore recommended that this behaviour is disabled at the SIIT gateways. See Section 4.4.

4. Implementation Requirements

[RFC6145] and [RFC6052] specifies the basic SIIT gateway. However, they specify some optional features that are very desirable when deploying SIIT in a data centre environment. This section list which additional features are required for an SIIT gateway optimised for a data centre environment.

4.1. Basic Requirements

The implementation MUST implement [RFC6145] with the algorithmic address mapping defined in [RFC6052]. It MUST NOT create any per-session state under any circumstance.

4.2. Static Address Mapping Function

The implementation MUST allow the operator to configure an arbitrary number of static 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 mapping would be:

- 198.51.0.1 <=> 2001:db8:12:34::c

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

- 198.51.0.1 => 2001:db8:12:34::c

- 2001:db8:12:34::c => 198.51.0.1

In both cases, if the gateway receives an IPv6 packet that has 2001:db8:12:34::c in either of the source and destination fields of the IP header, it MUST rewrite this field to 198.51.0.1 when translating to IPv4. Similarly, if the gateway receives an IPv4 packet that has 198.51.0.1 as the either the source or destination fields of the IP header, it MUST rewrite this field to 2001:db8:12:34::c. For all IPv4 or IPv6 source or destination field values for which there is no static mapping, [RFC6052] mapping MUST be used.

4.3. Support for Increasing the IPv6 Path MTU

In order to prevent unnecessary use of the IPv6 Fragmentation header, the implementation MUST support increasing the IPv6 Path MTU from its default value of 1280, as described in section 4 of [RFC6145].

4.4. Support for Disabling "Atomic Fragments"

The translator MUST provide a configuration function that allows the translator not to include the Fragment Header for non-fragmented IPv6 packets, cf. section 4 of [RFC6145].

4.5. Feature for Handling IPv4 Path MTUs Lower than 1260

In order to prevent unnecessary fragments, the implementation MUST support a feature which, if enabled by the operator, changes the translator's default behaviour accordingly:

- When translating an ICMPv4 Need To Fragment packet indicating a Path MTU smaller than or equal to 1260, the MTU field in the resulting ICMPv6 Packet Too Big is set to 1280.

- When translating an IPv6 packet that is smaller or equal to 1280 bytes, the DF flag in the resulting IPv4 packet is cleared, and an Identification value is generated. The translator MUST NOT generate any state as a result of this.

This is a modified version of the second approach described in section 6 of [RFC6145]. The default state of the feature SHOULD be disabled.

For the definition of an "Atomic Fragment", see [ATOMFRAG].

4.6. Loop Prevention Mechanism

As noted in Section 8.2, there is a potential for packets looping through the SIIT function if it receives an IPv4 packet for which there is no static 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 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.

5. Acknowledgements

TBD

6. 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].

7. IANA Considerations

This draft makes no request of the IANA.

8. Security Considerations

8.1. 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 come from 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.

8.2. Packets Looping Through the SIIT Function

The SIIT gateway receives an IPv4 packet destined to an address for which there is no static 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 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 mappings, or whose resulting IPv6 address does not have a more-specific route to the IPv6 network, are immediately discarded.

9. References

9.1. Normative References

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

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.

9.2. Informative References

[ATOMFRAG] Gont, F., "Processing of IPv6 "atomic" fragments", December 2011, <draft-gont-6man-ipv6-atomic-fragments-00>.

[FRAGDROP] Jaeggli, J., Colitti, L., Kumari, W., Vyncke, E., Kaeo, M., and T. Taylor, "Why Operators Filter Fragments and What It Implies", October 2012, <http://tools.ietf.org/

html/draft-taylor-v6ops-fragdrop-00>.

[RFC2373] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC2991] Thaler, D. and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection", RFC 2991, November 2000.

[RFC2993] Hain, T., "Architectural Implications of NAT", RFC 2993, November 2000.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC3235] Senie, D., "Network Address Translator (NAT)-Friendly Application Design Guidelines", RFC 3235, January 2002.

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

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, April 2011.

Author's Address

Tore Anderson
Redpill Linpro
Herregaardsveien 8B
NO-1168 Oslo
NORWAY

Phone: +47 959 31 212
Email: tore.anderson@redpill-linpro.com