Network Working Group
Internet-Draft
Updates: 6890 (if approved)
Intended status: Best Current Practice
Expires: November 3, 2017

R. Bonica
Juniper Networks
M. Cotton
ICANN
B. Haberman
Johns Hopkins University
L. Vegoda
ICANN

May 2, 2017

Updates to Special-Purpose IP Address Registries
draft-bchv-rfc6890bis-07

Abstract

This memo updates the IANA IPv4 and IPv6 Special-Purpose Address Registries to address issues raised by the definition of a "global" prefix. It also corrects several errors in registry entries to ensure the integrity of the IANA Special-Purpose Address Registries.

This memo updates RFC 6890.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In order to support new protocols and practices, the IETF occasionally reserves an address block for a special purpose. For example, [RFC1122] reserves an IPv4 address block (0.0.0.0/8) to represent the local (i.e., "this") network. Likewise, [RFC4291] reserves an IPv6 address block (fe80::/10) to represent link-scoped unicast addresses.

Several issues have been raised with the documentation of some of the special-purpose address blocks in [RFC6890]. Specifically, the definition of "global" provided in [RFC6890] was misleading as it slightly differed from the generally accepted definition of "global scope" (i.e., the ability to forward beyond the boundaries of an administrative domain, described as "global unicast" in the IPv6 addressing architecture [RFC4291]).

This memo updates the definition of "global" from [RFC6890] for the IPv4 and IPv6 Special-Purpose Address Registries, augments the fields contained within the registries in order to address the confusion raised by the definition of "global", and corrects some errors in some of the entries in the Special-Purpose Address Registries.

This memo updates [RFC6890].

2. IANA Considerations

2.1. Definition of Global

[RFC6890] defined the term "global" without taking into consideration the multiple uses of the term. Specifically, IP addresses can be global in terms of allocation scope as well as global in terms of routing/reachability. To address this ambiguity, the use of the term "global" defined in [RFC6890] is replaced with "globally reachable". The following definition replaces the definiton of "global" in the IANA Special-Purpose Address Registries:

- Globally Reachable - A boolean value indicating whether an IP datagram whose destination address is drawn from the allocated special-purpose address block is forwardable beyond a specified administrative domain.

The same relationship between the value of "Destination" and the values of "Forwardable" and "Global" described in [RFC6890] holds for "Globally Reachable". If the value of "Destination" is FALSE, the values of "Forwardable" and "Globally Reachable" must also be FALSE.

The "Global" column in the IPv4 Special-Purpose Address Registry (https://www.iana.org/assignments/iana-ipv4-special-registry) and the IPv6 Special-Purpose Address Registry (https://www.iana.org/assignments/iana-ipv6-special-registry) is renamed to "Globally Reachable".

2.2. Updates to the IPv4 Special-Purpose Address Registry

- Limited Broadcast prefix (255.255.255.255/32) - The Reserved-by-Protocol value is changed from False to True. This change is made to align the registry with reservation of the limited broadcast address with Section 7 of [RFC0919].

2.3. Updates to the IPv6 Special-Purpose Address Registry

The following changes to the IPv6 Special-Purpose Address Registry involves the insertion of two new footnotes. These changes require the footnotes to be re-numbered.

- TEREDO prefix (2001::/32) - The Globally Reachable value is changed from False to "N/A [2]". The [2] footnote states:

  * See Section 5 of [RFC4380] for details.

- EID Space for LISP (2001:5::/32) - All footnotes are incremented by 1.

- o 6to4 (2002::/16) - All footnotes are incremented by 1.

- o Unique-Local (fc00::/7) - The Globally Reachable value is changed from False to "False [7]". The [7] footnote states:

  * See [RFC4193] for more details on the routability of Unique-Local addresses. The Unique-Local prefix is drawn from the IPv6 Global Unicast Address range, but is specified as not globally routed.

3. Security Considerations

This document does not raise any security issues beyond those discussed in [RFC6890].

4. Acknowledgements

Brian Carpenter and C.M. Heard provided useful comments on initial versions of this document. Daniel Migault provided an in-depth review that helped strengthen the text within the document.

5. References

5.1. Normative References

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <http://www.rfc-editor.org/info/rfc6890>.

5.2. Informative References

[RFC0919] Mogul, J., "Broadcasting Internet Datagrams", STD 5, RFC 919, DOI 10.17487/RFC0919, October 1984, <http://www.rfc-editor.org/info/rfc919>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

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

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

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <http://www.rfc-editor.org/info/rfc4380>.

Authors' Addresses

Ronald Bonica
Juniper Networks

Email: rbonica@juniper.net

Michelle Cotton
ICANN

Email: michelle.cotton@icann.org

Brian Haberman
Johns Hopkins University

Email: brian@innovationslab.net

Leo Vegoda
ICANN

Email: leo.vegoda@icann.org

Routing Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

F. Baker
Cisco Systems
C. Bowers
Juniper Networks
J. Linkova
Google

October 31, 2016

Enterprise Multihoming using Provider-Assigned Addresses without Network Prefix Translation: Requirements and Solution

draft-bowbakova-rtgwg-enterprise-pa-multihoming-01

Abstract

Connecting an enterprise site to multiple ISPs using provider-assigned addresses is difficult without the use of some form of Network Address Translation (NAT). Much has been written on this topic over the last 10 to 15 years, but it still remains a problem without a clearly defined or widely implemented solution. Any multihoming solution without NAT requires hosts at the site to have addresses from each ISP and to select the egress ISP by selecting a source address for outgoing packets. It also requires routers at the site to take into account those source addresses when forwarding packets out towards the ISPs.

This document attempts to define a complete solution to this problem. It covers the behavior of routers to forward traffic taking into account source address, and it covers the behavior of host to select appropriate source addresses. It also covers any possible role that routers might play in providing information to hosts to help them select appropriate source addresses. In the process of exploring potential solutions, this documents also makes explicit requirements for how the solution would be expected to behave from the perspective of an enterprise site network administrator .

Status of This Memo

Table of Contents

1. Introduction

Site multihoming, the connection of a subscriber network to multiple upstream networks using redundant uplinks, is a common enterprise architecture for improving the reliability of its Internet connectivity. If the site uses provider-independent (PI) addresses, all traffic originating from the enterprise can use source addresses from the PI address space. Site multihoming with PI addresses is commonly used with both IPv4 and IPv6, and does not present any new technical challenges.

It may be desirable for an enterprise site to connect to multiple ISPs using provider-assigned (PA) addresses, instead of PI addresses. Multihoming with provider-assigned addresses is typically less expensive for the enterprise relative to using provider-independent addresses. PA multihoming is also a practice that should be facilitated and encouraged because it does not add to the size of the Internet routing table, whereas PI multihoming does. Note that PA is also used to mean "provider-aggregatable". In this document we assume that provider-assigned addresses are always provider-aggregatable.

With PA multihoming, for each ISP connection, the site is assigned a prefix from within an address block allocated to that ISP by its National or Regional Internet Registry. In the simple case of two ISPs (ISP-A and ISP-B), the site will have two different prefixes assigned to it (prefix-A and prefix-B). This arrangement is problematic. First, packets with the "wrong" source address may be dropped by one of the ISPs. In order to limit denial of service attacks using spoofed source addresses, BCP38 [RFC2827] recommends that ISPs filter traffic from customer sites to only allow traffic with a source address that has been assigned by that ISP. So a packet sent from a multihomed site on the uplink to ISP-B with a source address in prefix-A may be dropped by ISP-B.

However, even if ISP-B does not implement BCP38 or ISP-B adds prefix-A to its list of allowed source addresses on the uplink from the multihomed site, two-way communication may still fail. If the packet with source address in prefix-A was sent to ISP-B because the uplink to ISP-A failed, then if ISP-B does not drop the packet and the packet reaches its destination somewhere on the Internet, the return packet will be sent back with a destination address in prefix-A. The return packet will be routed over the Internet to ISP-A, but it will not be delivered to the multihomed site because its link with ISP-A has failed. Two-way communication would require some arrangement for ISP-B to advertise prefix-A when the uplink to ISP-A fails.

Note that the same may be true with a provider that does not implement BCP 38, if his upstream provider does, or has no corresponding route. The issue is not that the immediate provider implements ingress filtering; it is that someone upstream does, or lacks a route.

With IPv4, this problem is commonly solved by using [RFC1918] private address space within the multi-homed site and Network Address Translation (NAT) or Network Address/Port Translation (NAPT) on the uplinks to the ISPs. However, one of the goals of IPv6 is to eliminate the need for and the use of NAT or NAPT. Therefore, requiring the use of NAT or NAPT for an enterprise site to multihome with provider-assigned addresses is not an attractive solution.

[RFC6296] describes a translation solution specifically tailored to meet the requirements of multi-homing with provider-assigned IPv6 addresses. With the IPv6-to-IPv6 Network Prefix Translation (NPTv6) solution, within the site an enterprise can use Unique Local Addresses [RFC4193] or the prefix assigned by one of the ISPs. As traffic leaves the site on an uplink to an ISP, the source address gets translated to an address within the prefix assigned by the ISP on that uplink in a predictable and reversible manner. [RFC6296] is currently classified as Experimental, and it has been implemented by several vendors. See Section 5.2, for more discussion of NPTv6.

This document defines routing requirements for enterprise multihoming using provider-assigned IPv6 addresses. We have made no attempt to write these requirements in a manner that is agnostic to potential solutions. Instead, this document focuses on the following general class of solutions.

Each host at the enterprise has multiple addresses, at least one from each ISP-assigned prefix. Each host, as discussed in Section 4.1 and [RFC6724], is responsible for choosing the source address applied to each packet it sends. A host SHOULD be able respond dynamically to the failure of an uplink to a given ISP by no longer sending packets with the source address corresponding to that ISP. Potential mechanisms for the communication of changes in the network to the host are Neighbor Discovery Router Advertisements, DHCPv6, and ICMPv6.

The routers in the enterprise network are responsible for ensuring that packets are delivered to the "correct" ISP uplink based on source address. This requires that at least some routers in the site network are able to take into account the source address of a packet when deciding how to route it. That is, some routers must be capable of some form of Source Address Dependent Routing (SADR), if only as described in [RFC3704]. At a minimum, the routers connected to the ISP uplinks (the site exit routers or SERs) must be capable of Source Address Dependent Routing. Expanding the connected domain of routers capable of SADR from the site exit routers deeper into the site network will generally result in more efficient routing of traffic with external destinations.

The document first looks in more detail at the enterprise networking environments in which this solution is expected to operate. It then discusses existing and proposed mechanisms for hosts to select the source address applied to packets. Finally, it looks at the requirements for routing that are needed to support these enterprise network scenarios and the mechanisms by which hosts are expected to select source addresses dynamically based on network state.

2. Enterprise Multihoming Requirements

2.1. Simple ISP Connectivity with Connected SERs

We start by looking at a scenario in which a site has connections to two ISPs, as shown in Figure 1. The site is assigned the prefix 2001:db8:0:a000::/52 by ISP-A and prefix 2001:db8:0:b000::/52 by ISP-B. We consider three hosts in the site. H31 and H32 are on a LAN that has been assigned subnets 2001:db8:0:a010::/64 and 2001:db8:0:b010::/64. H31 has been assigned the addresses 2001:db8:0:a010::31 and 2001:db8:0:b010::31. H32 has been assigned 2001:db8:0:a010::32 and 2001:db8:0:b010::32. H41 is on a different subnet that has been assigned 2001:db8:0:a020::/64 and 2001:db8:0:b020::/64.

```
                                         2001:db8:0:1234::101   H101
                                                                  |
                                                                  |
 2001:db8:0:a010::31                                          --------
 2001:db8:0:b010::31                            ,-----.      /        \
                    +--+   +--+       +----+  ,'       `.   :          :
                +---|R1|---|R4|---+---|SERa|-+   ISP-A   +--+--        :
           H31--+   +--+   +--+   |   +----+  `.       ,'   :          :
                |                 |             `-----'     : Internet :
                |                 |                         :          :
                |                 |                         :          :
                |                 |                         :          :
                |                 |             ,-----.     :          :
           H32--+   +--+          |   +----+  ,'       `.   :          :
                +---|R2|----------+---|SERb|-+   ISP-B   +--+--        :
                    +--+          |   +----+  `.       ,'   :          :
                                  |             `-----'     :          :
                                  |                         :          :
                    +--+  +--+  +--+                         \        /
           H41------|R3|--|R5|--|R6|                          --------
                    +--+  +--+  +--+

 2001:db8:0:a020::41
 2001:db8:0:b020::41
```

Figure 1: Simple ISP Connectivity With Connected SERs

We refer to a router that connects the site to an ISP as a site edge router(SER). Several other routers provide connectivity among the internal hosts (H31, H32, and H41), as well as connecting the internal hosts to the Internet through SERa and SERb. In this example SERa and SERb share a direct connection to each other. In Section 2.2, we consider a scenario where this is not the case.

For the moment, we assume that the hosts are able to make good choices about which source addresses through some mechanism that doesn't involve the routers in the site network. Here, we focus on primary task of the routed site network, which is to get packets efficiently to their destinations, while sending a packet to the ISP that assigned the prefix that matches the source address of the packet. In Section 4, we examine what role the routed network may play in helping hosts make good choices about source addresses for packets.

With this solution, routers will need form of Source Address Dependent Routing, which will be new functionality. It would be useful if an enterprise site does not need to upgrade all routers to support the new SADR functionality in order to support PA multi-homing. We consider if this is possible and what are the tradeoffs of not having all routers in the site support SADR functionality.

In the topology in Figure 1, it is possible to support PA multihoming with only SERa and SERb being capable of SADR. The other routers can continue to forward based only on destination address, and exchange routes that only consider destination address. In this scenario, SERa and SERb communicate source-scoped routing information across their shared connection. When SERa receives a packet with a source address matching prefix 2001:db8:0:b000::/52 , it forwards the packet to SERb, which forwards it on the uplink to ISP-B. The analogous behaviour holds for traffic that SERb receives with a source address matching prefix 2001:db8:0:a000::/52.

In Figure 1, when only SERa and SERb are capable of source address dependent routing, PA multi-homing will work. However, the paths over which the packets are sent will generally not be the shortest paths. The forwarding paths will generally be more efficient as more routers are capable of SADR. For example, if R4, R2, and R6 are upgraded to support SADR, then can exchange source-scoped routes with SERa and SERb. They will then know to send traffic with a source address matching prefix 2001:db8:0:b000::/52 directly to SERb, without sending it to SERa first.

2.2. Simple ISP Connectivity Where SERs Are Not Directly Connected

In Figure 2, we modify the topology slightly by inserting R7, so that SERa and SERb are no longer directly connected. With this topology, it is not enough to just enable SADR routing on SERa and SERb to support PA multi-homing. There are two solutions to ways to enable PA multihoming in this topology.

```
                                         2001:db8:0:1234::101    H101
                                                                  |
                                                                  |
 2001:db8:0:a010::31                                          --------
 2001:db8:0:b010::31                            ,-----.      /        \
                    +--+   +--+       +----+  ,'       `.   :          :
                +---|R1|---|R4|---+---|SERa|-+   ISP-A   +--+--        :
           H31--+   +--+   +--+   |   +----+  `.       ,'   :          :
                |                 |             `-----'     : Internet :
                |               +--+                        :          :
                |               |R7|                        :          :
                |               +--+                        :          :
                |                 |             ,-----.     :          :
           H32--+   +--+          |   +----+  ,'       `.   :          :
                +---|R2|----------+---|SERb|-+   ISP-B   +--+--        :
                    +--+          |   +----+  `.       ,'   :          :
                                  |             `-----'     :          :
                                  |                         :          :
                    +--+  +--+  +--+                         \        /
           H41------|R3|--|R5|--|R6|                          --------
                    +--+  +--+  +--+                              |
                                                                  |
 2001:db8:0:a020::41                     2001:db8:0:5678::501    H501
 2001:db8:0:b020::41
```

Figure 2: Simple ISP Connectivity Where SERs Are Not Directly Connected

One option is to effectively modify the topology by creating a logical tunnel between SERa and SERb, using GRE for example. Although SERa and SERb are not directly connected physically in this topology, they can be directly connected logically by a tunnel.

The other option is to enable SADR functionality on R7. In this way, R7 will exchange source-scoped routes with SERa and SERb, making the three routers act as a single SADR domain. This illustrates the basic principle that the minimum requirement for the routed site network to support PA multi-homing is having all of the site exit routers be part of a connected SADR domain. Extending the connected SADR domain beyond that point can produce more efficient forwarding paths.

2.3. Enterprise Network Operator Expectations

Before considering a more complex scenario, let's look in more detail at the reasonably simple multihoming scenario in Figure 2 to understand what can reasonably be expected from this solution. As a general guiding principle, we assume an enterprise network operator will expect a multihomed network to behave as close as to a single-homed network as possible. So a solution that meets those expectations where possible is a good thing.

For traffic between internal hosts and traffic from outside the site to internal hosts, an enterprise network operator would expect there to be no visible change in the path taken by this traffic, since this traffic does not need to be routed in a way that depends on source address. It is also reasonable to expect that internal hosts should be able to communicate with each other using either of their source addresses without restriction. For example, H31 should be able to communicate with H41 using a packet with S=2001:db8:0:a010::31, D=2001:db8:0:b010::41, regardless of the state of uplink to ISP-B.

These goals can be accomplished by having all of the routers in the network continue to originate normal unscoped destination routes for their connected networks. If we can arrange so that these unscoped destination routes get used for forwarding this traffic, then we will have accomplished the goal of keeping forwarding of traffic destined for internal hosts, unaffected by the multihoming solution.

For traffic destined for external hosts, it is reasonable to expect that traffic with an source address from the prefix assigned by ISP-A to follow the path to that the traffic would follow if there is no connection to ISP-B. This can be accomplished by having SERa originate a source-scoped route of the form (S=2001:db8:0:a000::/52, D=::/0) . If all of the routers in the site support SADR, then the path of traffic exiting via ISP-A can match that expectation. If some routers don’t support SADR, then it is reasonable to expect that the path for traffic exiting via ISP-A may be different within the site. This is a tradeoff that the enterprise network operator may decide to make.

It is important to understand how this multihoming solution behaves when an uplink to one of the ISPs fails. To simplify this discussion, we assume that all routers in the site support SADR. We first start by looking at how the network operates when the uplinks to both ISP-A and ISP-B are functioning properly. SERa originates a source-scoped route of the form (S=2001:db8:0:a000::/52, D=::/0), and SERb is originates a source-scoped route of the form (S=2001:db8:0:b000::/52, D=::/0). These routes are distributed through the routers in the site, and they establish within the routers two set of forwarding paths for traffic leaving the site. One set of forwarding paths is for packets with source address in 2001:db8:0:a000::/52. The other set of forwarding paths is for packets with source address in 2001:db8:0:b000::/52. The normal destination routes which are not scoped to these two source prefixes play no role in the forwarding. Whether a packet exits the site via SERa or via SERb is completely determined by the source address applied to the packet by the host. So for example, when host H31 sends a packet to host H101 with (S=2001:db8:0:a010::31, D=2001:db8:0:1234::101), the packet will only be sent out the link from SERa to ISP-A.

Now consider what happens when the uplink from SERa to ISP-A fails. The only way for the packets from H31 to reach H101 is for H31 to start using the source address for ISP-B. H31 needs to send the following packet: (S=2001:db8:0:b010::31, D=2001:db8:0:1234::101).

This behavior is very different from the behavior that occurs with site multihoming using PI addresses or with PA addresses using NAT. In these other multi-homing solutions, hosts do not need to react to network failures several hops away in order to regain Internet access. Instead, a host can be largely unaware of the failure of an uplink to an ISP. When multihoming with PA addresses and NAT, existing sessions generally need to be re-established after a failure since the external host will receive packets from the internal host with a new source address. However, new sessions can be established without any action on the part of the hosts.

Another example where the behavior of this multihoming solution differs significantly from that of multihoming with PI address or with PA addresses using NAT is in the ability of the enterprise network operator to route traffic over different ISPs based on destination address. We still consider the fairly simple network of Figure 2 and assume that uplinks to both ISPs are functioning. Assume that the site is multihomed using PA addresses and NAT, and that SERa and SERb each originate a normal destination route for D=::/0, with the route origination dependent on the state of the uplink to the respective ISP.

Now suppose it is observed that an important application running between internal hosts and external host H101 experience much better performance when the traffic passes through ISP-A (perhaps because ISP-A provides lower latency to H101.) When multihoming this site with PI addresses or with PA addresses and NAT, the enterprise network operator can configure SERa to originate into the site network a normal destination route for D=2001:db8:0:1234::/64 (the destination prefix to reach H101) that depends on the state of the uplink to ISP-A. When the link to ISP-A is functioning, the destination route D=2001:db8:0:1234::/64 will be originated by SERa, so traffic from all hosts will use ISP-A to reach H101 based on the longest destination prefix match in the route lookup.

Implementing the same routing policy is more difficult with the PA multihoming solution described in this document since it doesn't use NAT. By design, the only way to control where a packet exits this network is by setting the source address of the packet. Since the network cannot modify the source address without NAT, the host must set it. To implement this routing policy, each host needs to use the source address from the prefix assigned by ISP-A to send traffic destined for H101. Mechanisms have been proposed to allow hosts to choose the source address for packets in a fine grained manner. We will discuss these proposals in Section 4. However, interacting with host operating systems in some manner to ensure a particular source address is chosen for a particular destination prefix is not what an enterprise network administrator would expect to have to do to implement this routing policy.

2.4. More complex ISP connectivity

The previous sections considered two variations of a simple multihoming scenario where the site is connected to two ISPs offering only Internet connectivity. It is likely that many actual enterprise multihoming scenarios will be similar to this simple example. However, there are more complex multihoming scenarios that we would like this solution to address as well.

It is fairly common for an ISP to offer a service in addition to Internet access over the same uplink. Two variation of this are reflected in Figure 3. In addition to Internet access, ISP-A offers a service which requires the site to access host H51 at 2001:db8:0:5555::51. The site has a single physical and logical connection with ISP-A, and ISP-A only allows access to H51 over that connection. So when H32 needs to access the service at H51 it needs to send packets with (S=2001:db8:0:a010::32, D=2001:db8:0:5555::51) and those packets need to be forward out the link from SERa to ISP-A.

```
                                         2001:db8:0:1234::101    H101
                                                                  |
                                                                  |
 2001:db8:0:a010::31                                          --------
 2001:db8:0:b010::31                            ,-----.      /        \
                    +--+   +--+       +----+  ,'       `.   :          :
                +---|R1|---|R4|---+---|SERa|-+   ISP-A   +--+--        :
           H31--+   +--+   +--+   |   +----+  `.       ,'   :          :
                |                 |             `-----'     : Internet :
                |                 |                |        :          :
                |                 |               H51       :          :
                |                 |     2001:db8:0:5555::51 :          :
                |               +--+                        :          :
                |               |R7|                        :          :
                |               +--+                        :          :
                |                 |                         :          :
                |                 |             ,-----.     :          :
           H32--+   +--+          |  +-----+  ,'       `.   :          :
                +---|R2|-----+----+--|SERb1|-+   ISP-B   +--+--        :
                    +--+     |       +-----+  `.       ,'   :          :
                           +--+                 `--|--'     :          :
  2001:db8:0:a010::32      |R8|                    |         \        /
                           +--+                 ,--|--.       --------
                             |       +-----+  ,'       `.        |
                             +-------|SERb2|-+   ISP-B   |       |
                             |       +-----+  `.       ,'       H501
                             |                  `-----'  2001:db8:0:5678
                             |                     |               ::501
                     +--+  +--+                   H61
            H41------|R3|--|R5|           2001:db8:0:6666::61
                     +--+  +--+

 2001:db8:0:a020::41
 2001:db8:0:b020::41
```

Figure 3: Internet access and services offered by ISP-A and ISP-B

ISP-B illustrates a variation on this scenario. In addition to Internet access, ISP-B also offers a service which requires the site to access host H61. The site has two connections to two different parts of ISP-B (shown as SERb1 and SERb2 in Figure 3). ISP-B expects Internet traffic to use the uplink from SERb1, while it expects it expects traffic destined for the service at H61 to use the uplink from SERb2. For either uplink, ISP-B expects the ingress traffic to have a source address matching the prefix it assigned to the site, 2001:db8:0:b000::/52.

As discussed before, we rely completely on the internal host to set the source address of the packet properly. In the case of a packet sent by H31 to access the service in ISP-B at H61, we expect the packet to have the following addresses: (S=2001:db8:0:b010::31, D=2001:db8:0:6666::61). The routed network has two potential ways of distributing routes so that this packet exits the site on the uplink at SERb2.

We could just rely on normal destination routes, without using source-prefix scoped routes. If we have SERb2 originate a normal unscoped destination route for D=2001:db8:0:6666::/64, the packets from H31 to H61 will exit the site at SERb2 as desired. We should not have to worry about SERa needing to originate the same route, because ISP-B should choose a globally unique prefix for the service at H61.

The alternative is to have SERb2 originate a source-prefix-scoped destination route of the form (S=2001:db8:0:b000::/52, D=2001:db8:0:6666::/64). From a forwarding point of view, the use of the source-prefix-scoped destination route would result in traffic with source addresses corresponding only to ISP-B being sent to SERb2. Instead, the use of the unscoped destination route would result in traffic with source addresses corresponding to ISP-A and ISP-B being sent to SERb2, as long as the destination address matches the destination prefix. It seems like either forwarding behavior would be acceptable.

However, from the point of view of the enterprise network administrator trying to configure, maintain, and trouble-shoot this multihoming solution, it seems much clearer to have SERb2 originate the source-prefix-scoped destination route correspond to the service offered by ISP-B. In this way, all of the traffic leaving the site is determined by the source-prefix-scoped routes, and all of the traffic within the site or arriving from external hosts is determined by the unscoped destination routes. Therefore, for this multihoming solution we choose to originate source-prefix-scoped routes for all traffic leaving the site.

2.5. ISPs and Provider-Assigned Prefixes

While we expect that most site multihoming involves connecting to only two ISPs, this solution allows for connections to an arbitrary number of ISPs to be supported. However, when evaluating scalable implementations of the solution, it would be reasonable to assume that the maximum number of ISPs that a site would connect to is five.

It is also useful to note that the prefixes assigned to the site by different ISPs will not overlap. This must be the case , since the provider-assigned addresses have to be globally unique.

2.6. Simplified Topologies

The topologies of many enterprise sites using this multihoming solution may in practice be simpler than the examples that we have used. The topology in Figure 1 could be further simplified by having all hosts directly connected to the LAN connecting the two site exit routers, SERa and SERb. The topology could also be simplified by having the uplinks to ISP-A and ISP-B both connected to the same site exit router. However, it is the aim of this draft to provide a solution that applies to a broad a range of enterprise site network topologies, so this draft focuses on providing a solution to the more general case. The simplified cases will also be supported by this solution, and there may even be optimizations that can be made for simplified cases. This solution however needs to support more complex topologies.

We are starting with the basic assumption that enterprise site networks can be quite complex from a routing perspective. However, even a complex site network can be multihomed to different ISPs with PA addresses using IPv4 and NAT. It is not reasonable to expect an enterprise network operator to change the routing topology of the site in order to deploy IPv6.

3. Generating Source-Prefix-Scoped Forwarding Tables

So far we have described in general terms how the routers in this solution that are capable of Source Address Dependent Routing will forward traffic using both normal unscoped destination routes and source-prefix-scoped destination routes. Here we give a precise method for generating a source-prefix-scoped forwarding table on a router that supports SADR.

1. Compute the next-hops for the source-prefix-scoped destination prefixes using only routers in the connected SADR domain. These are the initial source-prefix-scoped forwarding table entries.

2. Compute the next-hops for the unscoped destination prefixes using all routers in the IGP. This is the unscoped forwarding table.

3. Augment each source-prefix-scoped forwarding table with unscoped forwarding table entries based on the following rule. If the destination prefix of the unscoped forwarding entry exactly matches the destination prefix of an existing source-prefix-scoped forwarding entry (including destination prefix length),

then do not add the unscoped forwarding entry. If the destination prefix does NOT match an existing entry, then add the entry to the source-prefix-scoped forwarding table.

The forward tables produced by this process are used in the following way to forward packets.

1. If the source address of the packet matches one of the source prefixes, then look up the destination address of the packet in the corresponding source-prefix-scoped forwarding table to determine the next-hop for the packet.

2. If the source address of the packet does NOT match one of the source prefixes, then look up the destination address of the packet in unscoped forwarding table to determine the next-hop for the packet.

The following example illustrates how this process is used to create a forwarding table for each provider-assigned source prefix. We consider the multihomed site network in Figure 3. Initially we assume that all of the routers in the site network support SADR. Figure 4 shows the routes that are originated by the routers in the site network.

```
Routes originated by SERa:
(S=2001:db8:0:a000::/52, D=2001:db8:0:5555/64)
(S=2001:db8:0:a000::/52, D=::/0)
(D=2001:db8:0:5555::/64)
(D=::/0)

Routes originated by SERb1:
(S=2001:db8:0:b000::/52, D=::/0)
(D=::/0)

Routes originated by SERb2:
(S=2001:db8:0:b000::/52, D=2001:db8:0:6666::/64)
(D=2001:db8:0:6666::/64)

Routes originated by R1:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R2:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R3:
(D=2001:db8:0:a020::/64)
(D=2001:db8:0:b020::/64)
```

Figure 4: Routes Originated by Routers in the Site Network

Each SER originates destination routes which are scoped to the source prefix assigned by the ISP that the SER connects to. Note that the SERs also originate the corresponding unscoped destination route. This is not needed when all of the routers in the site support SADR. However, it is required when some routers do not support SADR. This will be discussed in more detail later.

We focus on how R8 constructs its source-prefix-scoped forwarding tables from these route advertisements. R8 computes the next hops for destination routes which are scoped to the source prefix 2001:db8:0:a000::/52. The results are shown in the first table in Figure 5. (In this example, the next hops are computed assuming that all links have the same metric.) Then, R8 computes the next hops for destination routes which are scoped to the source prefix 2001:db8:0:b000::/52. The results are shown in the second table in Figure 5 . Finally, R8 computes the next hops for the unscoped destination prefixes. The results are shown in the third table in Figure 5.

```
forwarding entries scoped to
source prefix = 2001:db8:0:a000::/52
============================================
D=2001:db8:0:5555/64      NH=R7
D=::/0                    NH=R7

forwarding entries scoped to
source prefix = 2001:db8:0:b000::/52
============================================
D=2001:db8:0:6666/64      NH=SERb2
D=::/0                    NH=SERb1

unscoped forwarding entries
============================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=SERb1
```

Figure 5: Forwarding Entries Computed at R8

The final step is for R8 to augment the source-prefix-scoped forwarding entries with unscoped forwarding entries. If an unscoped forwarding entry has the exact same destination prefix as an source-prefix-scoped forwarding entry (including destination prefix length), then the source-prefix-scoped forwarding entry wins.

As as an example of how the source scoped forwarding entries are augmented with unscoped forwarding entries, we consider how the two entries in the first table in Figure 5 (the table for source prefix = 2001:db8:0:a000::/52) are augmented with entries from the third table in Figure 5 (the table of unscoped forwarding entries). The first four unscoped forwarding entries (D=2001:db8:0:a010::/64, D=2001:db8:0:b010::/64, D=2001:db8:0:a020::/64, and D=2001:db8:0:b020::/64) are not an exact match for any of the existing entries in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, these four entries are added to the final forwarding table for source prefix 2001:db8:0:a000::/52. The result of adding these entries is reflected in first four entries the first table in Figure 6.

The next unscoped forwarding table entry is for D=2001:db8:0:5555::/64. This entry is an exact match for the existing entry in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, we do not replace the existing entry with the entry from the unscoped forwarding table. This is reflected in the fifth entry in the first table in Figure 6. (Note that since both scoped and unscoped entries have R7 as the next hop, the result of applying this rule is not visible.)

The next unscoped forwarding table entry is for D=2001:db8:0:6666::/64. This entry is not an exact match for any existing entries in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, we add this entry. This is reflected in the sixth entry in the first table in Figure 6.

The next unscoped forwarding table entry is for D=::/0. This entry is an exact match for the existing entry in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, we do not overwrite the existing source-prefix-scoped entry, as can be seen in the last entry in the first table in Figure 6.

```
if source address matches 2001:db8:0:a000::/52
then use this forwarding table
============================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=R7

else if source address matches 2001:db8:0:b000::/52
then use this forwarding table
============================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=SERb1

else use this forwarding table
============================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=SERb1
```

Figure 6: Complete Forwarding Tables Computed at R8

The forwarding tables produced by this process at R8 have the desired properties. A packet with a source address in 2001:db8:0:a000::/52 will be forwarded based on the first table in Figure 6. If the packet is destined for the Internet at large or the service at D=2001:db8:0:5555/64, it will be sent to R7 in the direction of SERa. If the packet is destined for an internal host, then the first four entries will send it to R2 or R5 as expected. Note that if this packet has a destination address corresponding to the service offered by ISP-B (D=2001:db8:0:5555::/64), then it will get forwarded to SERb2. It will be dropped by SERb2 or by ISP-B, since it the packet has a source address that was not assigned by ISP-B. However, this is expected behavior. In order to use the service offered by ISP-B, the host needs to originate the packet with a source address assigned by ISP-B.

In this example, a packet with a source address that doesn't match 2001:db8:0:a000::/52 or 2001:db8:0:b000::/52 must have originated from an external host. Such a packet will use the unscoped forwarding table (the last table in Figure 6). These packets will flow exactly as they would in absence of multihoming.

We can also modify this example to illustrate how it supports deployments where not all routers in the site support SADR. Continuing with the topology shown in Figure 3, suppose that R3 and R5 do not support SADR. Instead they are only capable of understanding unscoped route advertisements. The SADR routers in the network will still originate the routes shown in Figure 4. However, R3 and R5 will only understand the unscoped routes as shown in Figure 7.

```
Routes originated by SERa:
(D=2001:db8:0:5555::/64)
(D=::/0)

Routes originated by SERb1:
(D=::/0)

Routes originated by SERb2:
(D=2001:db8:0:6666::/64)

Routes originated by R1:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R2:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R3:
(D=2001:db8:0:a020::/64)
(D=2001:db8:0:b020::/64)
```

Figure 7: Routes Advertisements Understood by Routers that do no Support SADR

With these unscoped route advertisements, R5 will produce the forwarding table shown in Figure 8.

```
forwarding table
============================================
D=2001:db8:0:a010::/64    NH=R8
D=2001:db8:0:b010::/64    NH=R8
D=2001:db8:0:a020::/64    NH=R3
D=2001:db8:0:b020::/64    NH=R3
D=2001:db8:0:5555::/64    NH=R8
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=R8
```

Figure 8: Forwarding Table For R5, Which Doesn't Understand Source-Prefix-Scoped Routes

Any traffic that needs to exit the site will eventually hit a SADR-capable router. Once that traffic enters the SADR-capable domain, then it will not leave that domain until it exits the site. This property is required in order to guarantee that there will not be routing loops involving SADR-capable and non-SADR-capable routers.

Note that the mechanism described here for converting source-prefix-scoped destination prefix routing advertisements into forwarding state is somewhat different from that proposed in [I-D.ietf-rtgwg-dst-src-routing]. The method described in this document is intended to be easy to understand for network enterprise operators while at the same time being functionally correct. Another difference is that the method in this document assumes that source prefix will not overlap. Other differences between the two approaches still need to be understood and reconciled.

An interesting side-effect of deploying SADR is if all routers in a given network support SADR and have a scoped forwarding table, then the unscoped forwarding table can be eliminated which ensures that packets with legitimate source addresses only can leave the network (as there are no scoped forwarding tables for spoofed/bogon source addresses). It would prevent accidental leaks of ULA/reserved/link-local sources to the Internet as well as ensures that no spoofing is possible from the SADR-enabled network.

4. Mechanisms For Hosts To Choose Good Source Addresses In A Multihomed Site

Until this point, we have made the assumption that hosts are able to choose the correct source address using some unspecified mechanism. This has allowed us to just focus on what the routers in a multihomed site network need to do in order to forward packets to the correct ISP based on source address. Now we look at possible mechanisms for hosts to choose the correct source address. We also look at what role, if any, the routers may play in providing information that helps hosts to choose source addresses.

Any host that needs to be able to send traffic using the uplinks to a given ISP is expected to be configured with an address from the prefix assigned by that ISP. The host will control which ISP is used for its traffic by selecting one of the addresses configured on the host as the source address for outgoing traffic. It is the responsibility of the site network to ensure that a packet with the source address from an ISP is not sent on an uplink to that ISP.

If all of the ISP uplinks are working, the choice of source address by the host may be driven by the desire to load share across ISP uplinks, or it may be driven by the desire to take advantage of certain properties of a particular uplink or ISP. If any of the ISP uplinks is not working, then the choice of source address by the host can determine if packets get dropped.

How a host should make good decisions about source address selection in a multihomed site is not a solved problem. We do not attempt to solve this problem in this document. Instead we discuss the current state of affairs with respect to standardized solutions and implementation of those solutions. We also look at proposed solutions for this problem.

An external host initiating communication with a host internal to a PA multihomed site will need to know multiple addresses for that host in order to communicate with it using different ISPs to the multihomed site. These addresses are typically learned through DNS. (For simplicity, we assume that the external host is single-homed.) The external host chooses the ISP that will be used at the remote multihomed site by setting the destination address on the packets it transmits. For a sessions originated from an external host to an internal host, the choice of source address used by the internal host is simple. The internal host has no choice but to use the destination address in the received packet as the source address of the transmitted packet.

For a session originated by a host internal to the multi-homed site, the decision of what source address to select is more complicated. We consider three main methods for hosts to get information about the network. The two proactive methods are Neighbor Discovery Router Advertisements and DHCPv6. The one reactive method we consider is ICMPv6. Note that we are explicitly excluding the possibility of having hosts participate in or even listen directly to routing protocol advertisements.

First we look at how a host is currently expected to select the source and destination address with which it sends a packet.

4.1. Source Address Selection Algorithm on Hosts

[RFC6724] defines the algorithms that hosts are expected to use to select source and destination addresses for packets. It defines an algorithm for selecting a source address and a separate algorithm for selecting a destination address. Both of these algorithms depend on a policy table. [RFC6724] defines a default policy which produces certain behavior.

The rules in the two algorithms in [RFC6724] depend on many different properties of addresses. While these are needed for understanding how a host should choose addresses in an arbitrary environment, most of the rules are not relevant for understanding how a host should choose among multiple source addresses in mulitihomed envinronment when sending a packet to a remote host. Returning to the example in Figure 3, we look at what the default algorithms in [RFC6724] say about the source address that internal host H31 should use to send traffic to external host H101, somewhere on the Internet. Let’s look at what rules in [RFC6724] are actually used by H31 in this case.

There is no choice to be made with respect to destination address. H31 needs to send a packet with D=2001:db8:0:1234::101 in order to reach H101. So H31 have to choose between using S=2001:db8:0:a010::31 or S=2001:db8:0:b010::31 as the source address for this packet. We go through the rules for source address selection in Section 5 of [RFC6724]. Rule 1 (Prefer same address) is not useful to break the tie between source addresses, because neither the candidate source addresses equals the destination address. Rule 2 (Prefer appropriate scope) is also not used in this scenario, because both source addresses and the destination address have global scope.

Rule 3 (Avoid deprecated addresses) applies to an address that has been autoconfigured by a host using stateless address autoconfiguration as defined in [RFC4862]. An address autoconfigured by a host has a preferred lifetime and a valid lifetime. The address is preferred until the preferred lifetime expires, after which it becomes deprecated. A deprecated address can still be used, but it is better to use a preferred address. When the valid lifetime expires, the address cannot be used at all. The preferred and valid lifetimes for an autoconfigured address are set based on the corresponding lifetimes in the Prefix Information Option in Neighbor Discovery Router Advertisements. So a possible tool to control source address selection in this scenario would be to a host to make an address deprecated by having routers on that link, R1 and R2 in

Figure 3, send Prefix Information Option messages with the preferred lifetime for the source prefix to be discouraged (or prohibited) set to zero. This is a rather blunt tool, because it discourages or prohibits the use of that source prefix for all destinations. However, it may be useful in some scenarios.

Rule 4 (Avoid home addresses) does not apply here because we are not considering Mobile IP.

Rule 5 (Prefer outgoing interface) is not useful in this scenario, because both source addresses are assigned to the same interface.

Rule 5.5 (Prefer addresses in a prefix advertised by the next-hop) is not useful in the scenario when both R1 and R2 will advertise both source prefixes. However potentially this rule may allow a host to select the correct source prefix by selecting a next-hop. The most obvious way would be to make R1 to advertise itself as a default router and send PIO for 2001:db8:0:a010::/64, while R2 is advertising itself as a default router and sending PIO for 2001:db8:0:b010::/64. We'll discuss later how Rule 5.5 can be used to influence a source address selection in single-router topologies (e.g. when H41 is sending traffic using R3 as a default gateway).

Rule 6 (Prefer matching label) refers to the Label value determined for each source and destination prefix as a result of applying the policy table to the prefix. With the default policy table defined in Section 2.1 of [RFC6724], Label(2001:db8:0:a010::31) = 5, Label(2001:db8:0:b010::31) = 5, and Label(2001:db8:0:1234::101) = 5. So with the default policy, Rule 6 does not break the tie. However, the algorithms in [RFC6724] are defined in such as way that non-default address selection policy tables can be used. [RFC7078] defines a way to distribute a non-default address selection policy table to hosts using DHCPv6. So even though the application of rule 6 to this scenario using the default policy table is not useful, rule 6 may still be a useful tool.

Rule 7 (Prefer temporary addresses) has to do with the technique described in [RFC4941] to periodically randomize the interface portion of an IPv6 address that has been generated using stateless address autoconfiguration. In general, if H31 were using this technique, it would use it for both source addresses, for example creating temporary addresses 2001:db8:0:a010:2839:9938:ab58:830f and 2001:db8:0:b010:4838:f483:8384:3208, in addition to 2001:db8:0:a010::31 and 2001:db8:0:b010::31. So this rule would prefer the two temporary addresses, but it would not break the tie between the two source prefixes from ISP-A and ISP-B.

Rule 8 (Use longest matching prefix) dictates that between two candidate source addresses the one which has longest common prefix length with the destination address. For example, if H31 were selecting the source address for sending packets to H101, this rule would not be a tie breaker as for both candidate source addresses 2001:db8:0:a101::31 and 2001:db8:0:b101::31 the common prefix length with the destination is 48. However if H31 were selecting the source address for sending packets H41 address 2001:db8:0:a020::41, then this rule would result in using 2001:db8:0:a101::31 as a source (2001:db8:0:a101::31 and 2001:db8:0:a020::41 share the common prefix 2001:db8:0:a000::/58, while for ‘2001:db8:0:b101::31 and 2001:db8:0:a020::41 the common prefix is 2001:db8:0:a000::/51). Therefore rule 8 might be useful for selecting the correct source address in some but not all scenarios (for example if ISP-B services belong to 2001:db8:0:b000::/59 then H31 would always use 2001:db8:0:b010::31 to access those destinations).

So we can see that of the 8 source selection address rules from [RFC6724], five actually apply to our basic site multihoming scenario. The rules that are relevant to this scenario are summarized below.

o Rule 3: Avoid deprecated addresses.

o Rule 5.5: Prefer addresses in a prefix advertised by the next-hop.

o Rule 6: Prefer matching label.

o Rule 8: Prefer longest matching prefix.

The two methods that we discuss for controlling the source address selection through the four relevant rules above are SLAAC Router Advertisement messages and DHCPv6.

We also consider a possible role for ICMPv6 for getting traffic-driven feedback from the network. With the source address selection algorithm discussed above, the goal is to choose the correct source address on the first try, before any traffic is sent. However, another strategy is to choose a source address, send the packet, get feedback from the network about whether or not the source address is correct, and try another source address if it is not.

We consider four scenarios where a host needs to select the correct source address. The first is when both uplinks are working. The second is when one uplink has failed. The third one is a situation when one failed uplink has recovered. The last one is failure of both (all) uplinks.

4.2. Selecting Source Address When Both Uplinks Are Working

Again we return to the topology in Figure 3. Suppose that the site administrator wants to implement a policy by which all hosts need to use ISP-A to reach H01 at D=2001:db8:0:1234::101. So for example, H31 needs to select S=2001:db8:0:a010::31.

4.2.1. Distributing Address Selection Policy Table with DHCPv6

This policy can be implemented by using DHCPv6 to distribute an address selection policy table that assigns the same label to destination address that match 2001:db8:0:1234::/64 as it does to source addresses that match 2001:db8:0:a000::/52. The following two entries accomplish this.

Prefix	Precedence	Label
2001:db8:0:1234::/64	50	33
2001:db8:0:a000::/52	50	33

Figure 9: Policy table entries to implement a routing policy

This requires that the hosts implement [RFC6724], the basic source and destination address framework, along with [RFC7078], the DHCPv6 extension for distributing a non-default policy table. Note that it does NOT require that the hosts use DHCPv6 for address assignment. The hosts could still use stateless address autoconfiguration for address configuration, while using DHCPv6 only for policy table distribution (see [RFC3736]). However this method has a number of disadvantages:

- o DHCPv6 support is not a mandatory requirement for IPv6 hosts, so this method might not work for all devices.

- o Network administrators are required to explicitly configure the desired network access policies on DHCPv6 servers.

4.2.2. Controlling Source Address Selection With Router Advertisements

Neighbor Discovery currently has two mechanisms to communicate prefix information to hosts. The base specification for Neighbor Discovery (see [RFC4861]) defines the Prefix Information Option (PIO) in the Router Advertisement (RA) message. When a host receives a PIO with the A-flag set, it can use the prefix in the PIO as source prefix from which it assigns itself an IP address using stateless address autoconfiguration (SLAAC) procedures described in [RFC4862]. In the example of Figure 3, if the site network is using SLAAC, we would expect both R1 and R2 to send RA messages with PIOs for both source prefixes 2001:db8:0:a010::/64 and 2001:db8:0:b010::/64 with the

A-flag set. H31 would then use the SLAAC procedure to configure itself with the 2001:db8:0:a010::31 and 2001:db8:0:b010::31.

Whereas a host learns about source prefixes from PIO messages, hosts can learn about a destination prefix from a Router Advertisement containing Route Information Option (RIO), as specified in [RFC4191]. The destination prefixes in RIOs are intended to allow a host to choose the router that it uses as its first hop to reach a particular destination prefix.

As currently standardized, neither PIO nor RIO options contained in Neighbor Discovery Router Advertisements can communicate the information needed to implement the desired routing policy. PIO’s communicate source prefixes, and RIO communicate destination prefixes. However, there is currently no standardized way to directly associate a particular destination prefix with a particular source prefix.

[I-D.pfister-6man-sadr-ra] proposes a Source Address Dependent Route Information option for Neighbor Discovery Router Advertisements which would associate a source prefix and with a destination prefix. The details of [I-D.pfister-6man-sadr-ra] might need tweaking to address this use case. However, in order to be able to use Neighbor Discovery Router Advertisements to implement this routing policy, an extension that allows a R1 and R2 to explicitly communicate to H31 an association between S=2001:db8:0:a000::/52 D=2001:db8:0:1234::/64 would be needed.

However the Rule 5.5 of the source address selection (discussed above) together with default router preference (specified in [RFC4191]) and RIO can be used to influence a source address selection on a host as described below. Let’s look at source address selection on the host H41. It receives RAs from R3 with PIOs for 2001:db8:0:a020::/64 and 2001:db8:0:b020::/64. At that point all traffic would use the same next-hop (R3 link-local address) so Rule 5.5 does not apply. Now let’s assume that R3 supports SADR and has two scoped forwarding tables, one scoped to S=2001:db8:0:a000::/52 and another scoped to S=2001:db8:0:b000::/52. If R3 generates two different link-local addresses for its interface facing H41 (one for each scoped forwarding table, LLA_A and LLA_B) and starts sending two different RAs: one is sent from LLA_A and includes PIO for 2001:db8:0:a020::/64, another us sent from LLA_B and includes PIO for 2001:db8:0:b020::/64. Now it is possible to influence H41 source address selection for destinations which follow the default route by setting default router preference in RAs. If it is desired that H41 reaches H101 (or any destinations in the Internet) via ISP-A, then RAs sent from LLA_A should have default router preference set to 01 (high priority), while RAs sent from LLA_B should have preference set to 11 (low). Then LLA_A would be chosen as a next-hop for H101 and therefore (as per rule 5.5) 2001:db8:0:a020::41 would be selected as the source address. If, at the same time, it is desired that H61 is accessible via ISP-B then R3 should include a RIO for 2001:db8:0:6666::/64 to its RA sent from LLA_B. H41 would chose LLA_B as a next-hop for all traffic to H61 and then as per Rule 5.5, 2001:db8:0:b020::41 would be selected as a source address.

If in the above mentioned scenario it is desirable that all Internet traffic leaves the network via ISP-A and the link to ISP-B is used for accessing ISP-B services only (not as ISP-A link backup), then RAs sent by R3 from LLA_B should have Router Lifetime set to 0 and should include RIOs for ISP-B address space. It would instruct H41 to use LLA_A for all Internet traffic but use LLA_B as a next-hop while sending traffic to ISP-B addresses.

The proposed solution relies on SADR support by first-hop routers as well as SERs.

4.2.3. Controlling Source Address Selection With ICMPv6

We now discuss how one might use ICMPv6 to implement the routing policy to send traffic destined for H101 out the uplink to ISP-A, even when uplinks to both ISPs are working. If H31 started sending traffic to H101 with S=2001:db8:0:b010::31 and D=2001:db8:0:1234::101, it would be routed through SER-b1 and out the uplink to ISP-B. SERb1 could recognize that this is traffic is not following the desired routing policy and react by sending an ICMPv6 message back to H31.

In this example, we could arrange things so that SERb1 drops the packet with S=2001:db8:0:b010::31 and D=2001:db8:0:1234::101, and then sends to H31 an ICMPv6 Destination Unreachable message with Code 5 (Source address failed ingress/egress policy). When H31 receives this packet, it would then be expected to try another source address to reach the destination. In this example, H31 would then send a packet with S=2001:db8:0:a010::31 and D=2001:db8:0:1234::101, which will reach SERa and be forwarded out the uplink to ISP-A.

However, we would also want it to be the case that SERb1 does not enforce this routing policy when the uplink from SERa to ISP-A has failed. This could be accomplished by having SERa originate a source-prefix-scoped route for (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64) and have SERb1 monitor the presence of that route. If that route is not present (because SERa has stopped originating it), then SERb1 will not enforce the routing policy, and it will forward packets with S=2001:db8:0:b010::31 and D=2001:db8:0:1234::101 out its uplink to ISP-B.

We can also use this source-prefix-scoped route originated by SERa to communicate the desired routing policy to SERb1. We can define an EXCLUSIVE flag to be advertised together with the IGP route for (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64). This would allow SERa to communicate to SERb that SERb should reject traffic for D=2001:db8:0:1234::/64 and respond with an ICMPv6 Destination Unreachable Code 5 message, as long as the route for (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64) is present.

Finally, if we are willing to extend ICMPv6 to support this solution, then we could create a mechanism for SERb1 to tell the host what source address it should be using to successfully forward packets that meet the policy. In its current form, when SERb1 sends an ICMPv6 Destination Unreachable Code 5 message, it is basically saying, "This source address is wrong. Try another source address." It would be better is if the ICMPv6 message could say, "This source address is wrong. Instead use a source address in S=2001:db8:0:a000::/52."

However using ICMPv6 for signalling source address information back to hosts introduces new challenges. Most routers currently have software or hardware limits on generating ICMP messages. An site administrator deploying a solution that relies on the SERs generating ICMP messages could try to improve the performance of SERs for generating ICMP messages. However, in a large network, it is still likely that ICMP message generation limits will be reached. As a result hosts would not receive ICMPv6 back which in turns leads to traffic blackholing and poor user experience. To improve the scalability of ICMPv6-based signalling hosts SHOULD cache the preferred source address (or prefix) for the given destination. In addition, the same source prefix SHOULD be used for other destinations in the same /64 as the original destination address. The source prefix SHOULD have a specific lifetime. Expiration of the lifetime SHOULD trigger the source address selection algorithm again.

Using ICMPv6 Code 5 message for influencing source address selection allows an attacker to exhaust the list of candidate source addresses on the host by sending spoofed ICMPv6 Code 5 for all prefixes known on the network (therefore preventing a victim from establishing a communication with the destination host). To protect from such attack hosts SHOULD verify that the original packet header included into ICMPv6 error message was actually sent by the host.

4.2.4. Summary of Methods For Controlling Source Address Selection To Implement Routing Policy

So to summarize this section, we have looked at three methods for implementing a simple routing policy where all traffic for a given destination on the Internet needs to use a particular ISP, even when the uplinks to both ISPs are working.

The default source address selection policy cannot distinguish between the source addresses needed to enforce this policy, so a non-default policy table using associating source and destination prefixes using Label values would need to be installed on each host. A mechanism exists for DHCPv6 to distribute a non-default policy table but such solution would heavily rely on DHCPv6 support by host operating system. Moreover there is no mechanism to translate desired routing/traffic engineering policies into policy tables on DHCPv6 servers. Therefore using DHCPv6 for controlling address selection policy table is not recommended and SHOULD NOT be used.

At the same time Router Advertisements provide a reliable mechanism to influence source address selection process via PIO, RIO and default router preferences. As all those options have been standardized by IETF and are supported by various operating systems, no changes are required on hosts. First-hop routers in the enterprise network need to be able of sending different RAs for different SLAAC prefixes (either based on scoped forwarding tables or based on pre-configured policies).

SERs can enforce the routing policy by sending ICMPv6 Destination Unreachable messages with Code 5 (Source address failed ingress/egress policy) for traffic that is being sent with the wrong source address. The policy distribution can be automated by defining an EXCLUSIVE flag for the source-prefix-scoped route which can be set on the SER that originates the route. As ICMPv6 message generation can be rate-limited on routers, it SHOULD NOT be used as the only mechanism to influence source address selection on hosts. While hosts SHOULD select the correct source address for a given destination the network SHOULD signal any source address issues back to hosts using ICMPv6 error messages.

4.3. Selecting Source Address When One Uplink Has Failed

Now we discuss if DHCPv6, Neighbor Discovery Router Advertisements, and ICMPv6 can help a host choose the right source address when an uplink to one of the ISPs has failed. Again we look at the scenario in Figure 3. This time we look at traffic from H31 destined for external host H501 at D=2001:db8:0:5678::501. We initially assume that the uplink from SERa to ISP-A is working and that the uplink from SERb1 to ISP-B is working.

We assume there is no particular routing policy desired, so H31 is free to send packets with S=2001:db8:0:a010::31 or S=2001:db8:0:b010::31 and have them delivered to H501. For this example, we assume that H31 has chosen S=2001:db8:0:b010::31 so that the packets exit via SERb to ISP-B. Now we see what happens when the link from SERb1 to ISP-B fails. How should H31 learn that it needs to start sending the packet to H501 with S=2001:db8:0:a010::31 in order to start using the uplink to ISP-A? We need to do this in a way that doesn’t prevent H31 from still sending packets with S=2001:db8:0:b010::31 in order to reach H61 at D=2001:db8:0:6666::61.

4.3.1. Controlling Source Address Selection With DHCPv6

For this example we assume that the site network in Figure 3 has a centralized DHCP server and all routers act as DHCP relay agents. We assume that both of the addresses assigned to H31 were assigned via DHCP.

We could try to have the DHCP server monitor the state of the uplink from SERb1 to ISP-B in some manner and then tell H31 that it can no longer use S=2001:db8:0:b010::31 by settings its valid lifetime to zero. The DHCP server could initiate this process by sending a Reconfigure Message to H31 as described in Section 19 of [RFC3315]. Or the DHCP server can assign addresses with short lifetimes in order to force clients to renew them often.

This approach would prevent H31 from using S=2001:db8:0:b010::31 to reach the a host on the Internet. However, it would also prevent H31 from using S=2001:db8:0:b010::31 to reach H61 at D=2001:db8:0:6666::61, which is not desirable.

Another potential approach is to have the DHCP server monitor the uplink from SERb1 to ISP-B and control the choice of source address on H31 by updating its address selection policy table via the mechanism in [RFC7078]. The DHCP server could initiate this process by sending a Reconfigure Message to H31. Note that [RFC3315] requires that Reconfigure Message use DHCP authentication. DHCP authentication could be avoided by using short address lifetimes to force clients to send Renew messages to the server often. If the host is not obtaining its IP addresses from the DHCP server, then it would need to use the Information Refresh Time option defined in [RFC4242].

If the following policy table can be installed on H31 after the failure of the uplink from SERb1, then the desired routing behavior should be achieved based on source and destination prefix being matched with label values.

Prefix	Precedence	Label
::/0	50	44
2001:db8:0:a000::/52	50	44
2001:db8:0:6666::/64	50	55
2001:db8:0:b000::/52	50	55

Figure 10: Policy Table Needed On Failure Of Uplink From SERb1

The described solution has a number of significant drawbacks, some of them already discussed in Section 4.2.1.

- DHCPv6 support is not required for an IPv6 host and there are operating systems which do not support DHCPv6. Besides that, it does not appear that [RFC7078] has been widely implemented on host operating systems.

- [RFC7078] does not clearly specify this kind of a dynamic use case where address selection policy needs to be updated quickly in response to the failure of a link. In a large network it would present scalability issues as many hosts need to be reconfigured in very short period of time.

- No mechanism exists for making DHCPv6 servers aware of network topology/routing changes in the network. In general DHCPv6 servers monitoring network-related events sounds like a bad idea as completely new functionality beyond the scope of DHCPv6 role is required.

4.3.2. Controlling Source Address Selection With Router Advertisements

The same mechanism as discussed in Section 4.2.2 can be used to control the source address selection in the case of an uplink failure. If a particular prefix should not be used as a source for any destinations, then the router needs to send RA with Preferred Lifetime field for that prefix set to 0.

Let’s consider a scenario when all uplinks are operational and H41 receives two different RAs from R3: one from LLA_A with PIO for 2001:db8:0:a020::/64, default router preference set to 11 (low) and another one from LLA_B with PIO for 2001:db8:0:a020::/64, default router preference set to 01 (high) and RIO for 2001:db8:0:6666::/64. As a result H41 is using 2001:db8:0:b020::41 as a source address for all Internet traffic and those packets are sent by SERs to ISP-B. If SERb1 uplink to ISP-B failed, the desired behavior is that H41 stops using 2001:db8:0:b020::41 as a source address for all destinations but H61. To achieve that R3 should react to SERb1 uplink failure (which could be detected as the scoped route (S=2001:db8:0:b000::/52, D=::/0) disappearance) by withdrawing itself as a default router. R3 sends a new RA from LLA_B with Router Lifetime value set to 0 (which means that it should not be used as default router). That RA still contains PIO for 2001:db8:0:b020::/64 (for SLAAC purposes) and RIO for 2001:db8:0:6666::/64 so H41 can reach H61 using LLA_B as a next-hop and 2001:db8:0:b020::41 as a source address. For all traffic following the default route, LLA_A will be used as a next-hop and 2001:db8:0:a020::41 as a source address.

If all uplinks to ISP-B have failed and therefore source addresses from ISP-B address space should not be used at all, the forwarding table scoped S=2001:db8:0:b000::/52 contains no entries. Hosts can be instructed to stop using source addresses from that block by sending RAs containing PIO with Preferred Lifetime set to 0.

4.3.3. Controlling Source Address Selection With ICMPv6

Now we look at how ICMPv6 messages can provide information back to H31. We assume again that at the time of the failure H31 is sending packets to H501 using (S=2001:db8:0:b010::31, D=2001:db8:0:5678::501). When the uplink from SERb1 to ISP-B fails, SERb1 would stop originating its source-prefix-scoped route for the default destination (S=2001:db8:0:b000::/52, D=::/0) as well as its unscoped default destination route. With these routes no longer in the IGP, traffic with (S=2001:db8:0:b010::31, D=2001:db8:0:5678::501) would end up at SERa based on the unscoped default destination route being originated by SERa. Since that traffic has the wrong source address to be forwarded to ISP-A, SERa would drop it and send a Destination Unreachable message with Code 5 (Source address failed ingress/egress policy) back to H31. H31 would then know to use another source address for that destination and would try with (S=2001:db8:0:a010::31, D=2001:db8:0:5678::501). This would be forwarded to SERa based on the source-prefix-scoped default destination route still being originated by SERa, and SERa would forward it to ISP-A. As discussed above, if we are willing to extend ICMPv6, SERa can even tell H31 what source address it should use to reach that destination. The expected host behaviour has been discussed in Section 4.2.3. Potential issue with using ICMPv6 for signalling source address issues back to hosts is that uplink to an ISP-B failure immediately invalidates source addresses from 2001:db8:0:b000::/52 for all hosts which triggers a large number of ICMPv6 being sent back to hosts - the same scalability/rate limiting issues discussed in Section 4.2.3 would apply.

4.3.4. Summary Of Methods For Controlling Source Address Selection On The Failure Of An Uplink

It appears that DHCPv6 is not particularly well suited to quickly changing the source address used by a host in the event of the failure of an uplink, which eliminates DHCPv6 from the list of potential solutions. On the other hand Router Advertisements provides a reliable mechanism to dynamically provide hosts with a list of valid prefixes to use as source addresses as well as prevent particular prefixes to be used. While no additional new features are required to be implemented on hosts, routers need to be able to send RAs based on the state of scoped forwarding tables entries and to react to network topology changes by sending RAs with particular parameters set.

The use of ICMPv6 Destination Unreachable messages generated by the SER (or any SADR-capable) routers seem like they have the potential to provide a support mechanism together with RAs to signal source address selection errors back to hosts, however scalability issues may arise in large networks in case of sudden topology change. Therefore it is highly desirable that hosts are able to select the correct source address in case of uplinks failure with ICMPv6 being an additional mechanism to signal unexpected failures back to hosts.

The current behavior of different host operating system when receiving ICMPv6 Destination Unreachable message with code 5 (Source address failed ingress/egress policy) is not clear to the authors. Information from implementers, users, and testing would be quite helpful in evaluating this approach.

4.4. Selecting Source Address Upon Failed Uplink Recovery

The next logical step is to look at the scenario when a failed uplink on SERb1 to ISP-B is coming back up, so hosts can start using source addresses belonging to 2001:db8:0:b000::/52 again.

4.4.1. Controlling Source Address Selection With DHCPv6

The mechanism to use DHCPv6 to instruct the hosts (H31 in our example) to start using prefixes from ISP-B space (e.g. S=2001:db8:0:b010::31 for H31) to reach hosts on the Internet is quite similar to one discussed in Section 4.3.1 and shares the same drawbacks.

4.4.2. Controlling Source Address Selection With Router Advertisements

Let’s look at the scenario discussed in Section 4.3.2. If the uplink(s) failure caused the complete withdrawal of prefixes from 2001:db8:0:b000::/52 address space by setting Preferred Lifetime value to 0, then the recovery of the link should just trigger new RA being sent with non-zero Preferred Lifetime. In another scenario discussed in Section 4.3.2, the SERb1 uplink to ISP-B failure leads to disappearance of the (S=2001:db8:0:b000::/52, D=::/0) entry from the forwarding table scoped to S=2001:db8:0:b000::/52 and, in turn, caused R3 to send RAs from LLA_B with Router Lifetime set to 0. The recovery of the SERb1 uplink to ISP-B leads to (S=2001:db8:0:b000::/52, D=::/0) scoped forwarding entry re-appearance and instructs R3 that it should advertise itself as a default router for ISP-B address space domain (send RAs from LLA_B with non-zero Router Lifetime).

4.4.3. Controlling Source Address Selection With ICMP

It looks like ICMPv6 provides a rather limited functionality to signal back to hosts that particular source addresses have become valid again. Unless the changes in the uplink state a particular (S,D) pair, hosts can keep using the same source address even after an ISP uplink has come back up. For example, after the uplink from SERb1 to ISP-B had failed, H31 received ICMPv6 Code 5 message (as described in Section 4.3.3) and allegedly started using (S=2001:db8:0:a010::31, D=2001:db8:0:5678::501) to reach H501. Now when the SERb1 uplink comes back up, the packets with that (S,D) pair are still routed to SERa1 and sent to the Internet. Therefore H31 is not informed that it should stop using 2001:db8:0:a010::31 and start using 2001:db8:0:b010::31 again. Unless SERa has a policy configured to drop packets (S=2001:db8:0:a010::31, D=2001:db8:0:5678::501) and send ICMPv6 back if SERb1 uplink to ISP-B is up, H31 will be unaware of the network topology change and keep using S=2001:db8:0:a010::31 for Internet destinations, including H51.

One of the possible option may be using a scoped route with EXCLUSIVE flag as described in Section 4.2.3. SERa1 uplink recovery would cause (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64) route to reappear in the routing table. In the absence of that route packets to H101 which were sent to ISP-B (as ISP-A uplink was down) with source addresses from 2001:db8:0:b000::/52. When the route re-appears SERb1 would reject those packets and sends ICMPv6 back as discussed in Section 4.2.3. Practically it might lead to scalability issues which have been already discussed in Section 4.2.3 and Section 4.4.3.

4.4.4. Summary Of Methods For Controlling Source Address Selection Upon Failed Uplink Recovery

Once again DHCPv6 does not look like reasonable choice to manipulate source address selection process on a host in the case of network topology changes. Using Router Advertisement provides the flexible mechanism to dynamically react to network topology changes (if routers are able to use routing changes as a trigger for sending out RAs with specific parameters). ICMPv6 could be considered as a supporting mechanism to signal incorrect source address back to hosts but should not be considered as the only mechanism to control the address selection in multihomed environments.

4.5. Selecting Source Address When All Uplinks Failed

One particular tricky case is a scenario when all uplinks have failed. In that case there is no valid source address to be used for any external destinations while it might be desirable to have intra-site connectivity.

4.5.1. Controlling Source Address Selection With DHCPv6

From DHCPv6 perspective uplinks failure should be treated as two independent failures and processed as described in Section 4.3.1. At this stage it is quite obvious that it would result in quite complicated policy table which needs to be explicitly configured by administrators and therefore seems to be impractical.

4.5.2. Controlling Source Address Selection With Router Advertisements

As discussed in Section 4.3.2 an uplink failure causes the scoped default entry to disappear from the scoped forwarding table and triggers RAs with zero Router Lifetime. Complete disappearance of all scoped entries for a given source prefix would cause the prefix being withdrawn from hosts by setting Preferred Lifetime value to zero in PIO. If all uplinks (SERa, SERb1 and SERb2) failed, hosts either lost their default routers and/or have no global IPv6 addresses to use as a source. (Note that 'uplink failure' might mean 'IPv6 connectivity failure with IPv4 still being reachable', in which case hosts might fall back to IPv4 if there is IPv4 connectivity to destinations). As a results intra-site connectivity is broken. One of the possible way to solve it is to use ULAs.

All hosts have ULA addresses assigned in addition to GUAs and used for intra-site communication even if there is no GUA assigned to a host. To avoid accidental leaking of packets with ULA sources SADR-capable routers SHOULD have a scoped forwarding table for ULA source for internal routes but MUST NOT have an entry for D=::/0 in that table. In the absence of (S=ULA_Prefix; D=::/0) first-hop routers will send dedicated RAs from a unique link-local source LLA_ULA with PIO from ULA address space, RIO for the ULA prefix and Router Lifetime set to zero. The behaviour is consistent with the situation when SERb1 lost the uplink to ISP-B (so there is no Internet connectivity from 2001:db8:0:b000::/52 sources) but those sources can be used to reach some specific destinations. In the case of ULA there is no Internet connectivity from ULA sources but they can be used to reach another ULA destinations. Note that ULA usage could be particularly useful if all ISPs assign prefixes via DHCP-PD. In the absence of ULAs uplinks failure hosts would lost all their GUAs upon prefix lifetime expiration which again makes intra-site communication impossible.

4.5.3. Controlling Source Address Selection With ICMPv6

In case of all uplinks failure all SERs will drop outgoing IPv6 traffic and respond with ICMPv6 error message. In the large network when many hosts are trying to reach Internet destinations it means that SERs need to generate an ICMPv6 error to every packet they receive from hosts which presents the same scalability issues discussed in Section 4.3.3

4.5.4. Summary Of Methods For Controlling Source Address Selection When All Uplinks Failed

Again, combining SADR with Router Advertisements seems to be the most flexible and scalable way to control the source address selection on hosts.

4.6. Summary Of Methods For Controlling Source Address Selection

To summarize the scenarios and options discussed above:

While DHCPv6 allows administrators to manipulate source address selection policy tables, this method has a number of significant disadvantages which eliminates DHCPv6 from a list of potential solutions:

1. It required hosts to support DHCPv6 and its extension (RFC7078);

2. DHCPv6 server need to monitor network state and detect routing changes.

3. Network topology/routing policy changes could trigger simultaneous re-configuration of large number of hosts which present serious scalability issues.

The use of Router Advertisements to influence the source address selection on hosts seem to be the most reliable, flexible and scalable solution. It has the following benefits:

1. no new (non-standard) functionality needs to be implemented on hosts (except for [RFC4191] support);

2. no changes in RA format;

3. Routers can react to routing table changes by sending RAs which would minimize the failover time in the case of network topology changes;

4. information required for source address selection is broadcast to all affected hosts in case of topology change event which improves the scalability of the solution (comparing to DHCPv6 reconfiguration or ICMPv6 error messages).

To fully benefit from the RA-based solution, first-hop routers need to implement SADR and be able to send dedicated RAs per scoped forwarding table as discussed above, reacting to network changes with sending new RAs. It should be noted that the proposed solution would work even if first-hop routers are not SADR-capable but still able to send individual RAs for each ISP prefix and react to topology changes as discussed above

The RA-based solution relies heavily on hosts correctly implementing default address selection algorith as defined in [RFC6724] and in particular, Rule 5.5. There are some evidences that not all host OSes have that rule implemented currenly (it should be noted that [I-D.ietf-6man-multi-homed-host] states that Rule 5.5 SHOULD be implemented.

ICMPv6 Code 5 error message SHOULD be used to complement RA-based solution to signal incorrect source address selection back to hosts, but it SHOULD NOT be considered as the stand-alone solution. To prevent scenarios when hosts in multihomed envinronments incorrectly identify onlink/offlink destimations, hosts should treat ICMPv6 Redirects as discussed in [I-D.ietf-6man-multi-homed-host].

4.7. Other Configuration Parameters

4.7.1. DNS Configuration

In mutihomed envinronment each ISP might provide their own list of DNS servers. E.g. in the topology show on Figure 3, ISP-A might provide recursive DNS server H51 2001:db8:0:5555::51, while ISP-B might provide H61 2001:db8:0:6666::61 as a recursive DNS server. If the multihomed enterprise network is not running their own recursive resolver then hosts need to be configured with DNS server IPv6 addresses. [RFC6106] defines IPv6 Router Advertisement options to allow IPv6 routers to advertise a list of DNS recursive server addresses and a DNS Search List to IPv6 hosts. Using RDNSS together with 'scoped' RAs as described above would allow a first-hop router (R3 in the Figure 3) to send DNS server addresses and search lists provided by each ISPs.

As discussed in Section 4.5.2, failure of all ISP uplinks would cause depreaction of all addresses assigned to a host from ISPs address space. Most likely intra-site IPv6 connectivity would be still desirable so Section 4.5.2 proposes a usage of ULAs to enable intra-site communication. In such scenario the enterprise network should run its own recursive DNS server(s) and provide its ULA addresses to hosts via RDNSS mechanism in RAs send for ULA-scoped forwarding table as described in Section 4.5.2.

It should be noted that [RFC6106] explicitly prohibits using DNS information if the RA router Lifetime expired: "An RDNSS address or a DNSSL domain name MUST be used only as long as both the RA router Lifetime (advertised by a Router Advertisement message) and the corresponding option Lifetime have not expired.". Therefore hosts might ignore RDNSS information provided in ULA-scoped RAs as those RAs would have router lifetime set to 0. However the updated version of RFC6106 ([I-D.ietf-6man-rdnss-rfc6106bis]) has that requirement removed.

5. Other Solutions

5.1. Shim6

The Shim6 working group specified the Shim6 protocol [RFC5533] which allows a host at a multihomed site to communicate with an external host and exchange information about possible source and destination address pairs that they can use to communicate. It also specified the REAP protocol [RFC5534] to detect failures in the path between working address pairs and find new working address pairs. A fundamental requirement for Shim6 is that both internal and external hosts need to support Shim6. That is, both the host internal to the multihomed site and the host external to the multihomed site need to support Shim6 in order for there to be any benefit for the internal host to run Shim6. The Shim6 protocol specification was published in 2009, but it has not been implemented on widely used operating systems.

We do not consider Shim6 to be a viable solution. It suffers from the fact that it requires widespread deployment of Shim6 on hosts all over the Internet before the host at a PA multihomed site sees significant benefit. However, there appears to be no motivation for the vast majority of hosts on the Internet (which are not at PA multihomed sites) to deploy Shim6. This may help explain why Shim6 has not been widely implemented.

5.2. IPv6-to-IPv6 Network Prefix Translation

IPv6-to-IPv6 Network Prefix Translation (NPTv6) [RFC6296] is not the focus of this document. This document describes a solution where a host in a multihomed site determines which ISP a packet will be sent to based on the source address it applies to the packet. This solution has many moving parts. It requires some routers in the enterprise site to support some form of Source Address Dependent Routing (SADR). It requires a host to be able to learn when the uplink to an ISP fails so that it can stop using the source address corresponding to that ISP. Ongoing work to create mechanisms to accomplish this are discussed in this document, but they are still a work in progress.

This document attempts to create a PA multihoming solution that is as easy as possible for an enterprise to deploy. However, the success of this solution will depend greatly on whether or not the mechanisms for hosts to select source addresses based on the state of ISP uplinks gets implemented across a wide range of operating systems as the default mode of operation. Until that occurs, NPTv6 should still be considered a viable option to enable PA multihoming for enterprises.

6. IANA Considerations

This memo asks the IANA for no new parameters.

7. Security Considerations

7.1. Privacy Considerations

8. Acknowledgements

The original outline was suggested by Ole Troan.

9. References

9.1. Normative References

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <http://www.rfc-editor.org/info/rfc1123>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

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

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

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3582] Abley, J., Black, B., and V. Gill, "Goals for IPv6 Site-Multihoming Architectures", RFC 3582, DOI 10.17487/RFC3582, August 2003, <http://www.rfc-editor.org/info/rfc3582>.

[RFC4116] Abley, J., Lindqvist, K., Davies, E., Black, B., and V. Gill, "IPv4 Multihoming Practices and Limitations", RFC 4116, DOI 10.17487/RFC4116, July 2005, <http://www.rfc-editor.org/info/rfc4116>.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <http://www.rfc-editor.org/info/rfc4191>.

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

[RFC4218] Nordmark, E. and T. Li, "Threats Relating to IPv6 Multihoming Solutions", RFC 4218, DOI 10.17487/RFC4218, October 2005, <http://www.rfc-editor.org/info/rfc4218>.

[RFC4219] Lear, E., "Things Multihoming in IPv6 (MULTI6) Developers Should Think About", RFC 4219, DOI 10.17487/RFC4219, October 2005, <http://www.rfc-editor.org/info/rfc4219>.

[RFC4242] Venaas, S., Chown, T., and B. Volz, "Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 4242, DOI 10.17487/RFC4242, November 2005, <http://www.rfc-editor.org/info/rfc4242>.

[RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, DOI 10.17487/RFC6106, November 2010, <http://www.rfc-editor.org/info/rfc6106>.

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

[RFC7157] Troan, O., Ed., Miles, D., Matsushima, S., Okimoto, T., and D. Wing, "IPv6 Multihoming without Network Address Translation", RFC 7157, DOI 10.17487/RFC7157, March 2014, <http://www.rfc-editor.org/info/rfc7157>.

9.2. Informative References

[I-D.baker-ipv6-isis-dst-src-routing]
Baker, F. and D. Lamparter, "IPv6 Source/Destination Routing using IS-IS", draft-baker-ipv6-isis-dst-src-routing-06 (work in progress), October 2016.

[I-D.baker-rtgwg-src-dst-routing-use-cases]
Baker, F., Xu, M., Yang, S., and J. Wu, "Requirements and Use Cases for Source/Destination Routing", draft-baker-rtgwg-src-dst-routing-use-cases-02 (work in progress), April 2016.

[I-D.boutier-babel-source-specific]
Boutier, M. and J. Chroboczek, "Source-Specific Routing in Babel", draft-boutier-babel-source-specific-01 (work in progress), May 2015.

[I-D.huitema-shim6-ingress-filtering]
Huitema, C., "Ingress filtering compatibility for IPv6 multihomed sites", draft-huitema-shim6-ingress-filtering-00 (work in progress), September 2005.

[I-D.ietf-6man-multi-homed-host]
Baker, F. and B. Carpenter, "First-hop router selection by hosts in a multi-prefix network", draft-ietf-6man-multi-homed-host-10 (work in progress), October 2016.

[I-D.ietf-6man-rdnss-rfc6106bis]
Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", draft-ietf-6man-rdnss-rfc6106bis-14 (work in progress), June 2016.

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

[I-D.ietf-mptcp-experience]
Bonaventure, O., Paasch, C., and G. Detal, "Use Cases and Operational Experience with Multipath TCP", draft-ietf-mptcp-experience-07 (work in progress), October 2016.

[I-D.ietf-rtgwg-dst-src-routing]
Lamparter, D. and A. Smirnov, "Destination/Source Routing", draft-ietf-rtgwg-dst-src-routing-02 (work in progress), May 2016.

[I-D.pfister-6man-sadr-ra]
Pfister, P., "Source Address Dependent Route Information Option for Router Advertisements", draft-pfister-6man-sadr-ra-01 (work in progress), June 2015.

[I-D.xu-src-dst-bgp]
Xu, M., Yang, S., and J. Wu, "Source/Destination Routing Using BGP-4", draft-xu-src-dst-bgp-00 (work in progress), March 2016.

[PATRICIA]
Morrison, D., "Practical Algorithm to Retrieve Information Coded in Alphanumeric", Journal of the ACM 15(4) pp514-534, October 1968.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <http://www.rfc-editor.org/info/rfc3704>.

[RFC3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, DOI 10.17487/RFC3736, April 2004, <http://www.rfc-editor.org/info/rfc3736>.

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

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

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <http://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, DOI 10.17487/RFC5533, June 2009, <http://www.rfc-editor.org/info/rfc5533>.

[RFC5534] Arkko, J. and I. van Beijnum, "Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming", RFC 5534, DOI 10.17487/RFC5534, June 2009, <http://www.rfc-editor.org/info/rfc5534>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

[RFC7078] Matsumoto, A., Fujisaki, T., and T. Chown, "Distributing Address Selection Policy Using DHCPv6", RFC 7078, DOI 10.17487/RFC7078, January 2014, <http://www.rfc-editor.org/info/rfc7078>.

[RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April 2016, <http://www.rfc-editor.org/info/rfc7788>.

Appendix A. Change Log

Initial Version: July 2016

Authors' Addresses

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA

Email: fred@cisco.com

Chris Bowers
Juniper Networks
Sunnyvale, California 94089
USA

Email: cbowers@juniper.net

Jen Linkova
Google
Mountain View, California 94043
USA

Email: furry@google.com

Network Working Group B. Carpenter
Internet-Draft Univ. of Auckland
Intended status: Standards Track June 8, 2016
Expires: December 10, 2016

What does 'global' mean in IPv6?
draft-carpenter-6man-whats-global-00

Abstract

The word 'global' is used in two different ways in various IPv6-related RFCs and an IANA registry. This document describes the resulting problem.

Status of This Memo

Copyright Notice

Table of Contents

1. Problem description

As defined in the IPv6 Addressing Architecture [I-D.ietf-6man-rfc4291bis], most of the IPv6 address space is reserved for Global Unicast addresses. The high order bits of such addresses are named ’global routing prefix’. However, the word ’global’ is not itself defined in the context of unicast addresses.

One subset of Global Unicast address space is defined for Unique Local Addresses [RFC4193]. One can quarrel with something being called ’global’ and ’local’ at the same time, but RFC 4193 is categorical:

> This document defines an IPv6 unicast address format that is globally unique and is intended for local communications, usually inside of a site. These addresses are not expected to be routable on the global Internet.
> ...
> - Globally unique prefix (with high probability of uniqueness).
> ...
> - In practice, applications may treat these addresses like global scoped addresses.
> ...
> By default, the scope of these addresses is global. That is, they are not limited by ambiguity like the site-local addresses defined in [ADDARCH]. Rather, these prefixes are globally unique, and as such, their applicability is greater than site-local addresses. Their limitation is in the routability of the prefixes, which is limited to a site and any explicit routing agreements with other sites to propagate them...

In summary: ULAs are defined in these standards track documents as ’global’.

However, the IANA registry for special-purpose IPv6 addresses <http://www.iana.org/assignments/iana-ipv6-special-registry/iana-ipv6-special-registry.xhtml>, and the RFC that controls it [RFC6890] use the following definition:

o Global - A boolean value indicating whether an IP datagram whose destination address is drawn from the allocated special-purpose address block is forwardable beyond a specified administrative domain.

It is evident, even from the last sentence quoted above from RFC 4193, that ULAs do not meet this definition of 'global'. As a result, they are marked in the registry with Global = False. The registry also assigns them the property Forwardable = True, which is of course valid, but the fact remains that some RFCs say that ULAs are global, but RFC 6890 and the registry say that they are not.

This inconsistency has consequences. Of course, it is always possible for code that manipulates IPv6 addresses to determine with certainty that a given address is, or is not, a ULA. But any code that uses the property 'global' from the IANA registry as a decision criterion might be wrong.

As an example, consider the Python 'ipaddress' module <https://docs.python.org/3.4/library/ipaddress.html#ipaddress.IPv4Address.is_private>, which explicitly cites the IANA registry. It provides the property 'is_global' which tests False for ULAs. A reader of RFC 4193 would expect True. The correct test in Python (apart from an explicit match with fc00::/7) is (is_private and not is_link_local).

2. Possible fixes

1. Do nothing.

2. Change the registry entry for ULAs to Global=True (and update text and RFC 6890 accordingly).

3. That, plus rename the registry column from 'Global' to 'Global scope'.

4. Change the registry entry for ULAs to Global=Undefined (and update text and RFC 6890 accordingly).

5. Rename the registry column from 'Global' to 'Globally reachable' (and update text and RFC 6890 accordingly).

6. That, plus add a registry column for 'Global scope'.

7. Your suggestion goes here.

3. Security Considerations

Misclassification of a ULA as non-global might cause it to be used for a purpose that should be limited to link-local addresses for security reasons.

4. IANA Considerations

If any changes are made as a result of this discussion, they will require IANA actions.

5. Normative References

[I-D.ietf-6man-rfc4291bis]
Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", draft-ietf-6man-rfc4291bis-02 (work in progress), April 2016.

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

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <http://www.rfc-editor.org/info/rfc6890>.

Author’s Address

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

F. Gont
SI6 Networks / UTN-FRH
C. Huitema
Private Octopus Inc.
S. Krishnan
Ericsson Research
G. Gont
SI6 Networks
M. Garcia Corbo
SITRANS
March 24, 2018

Recommendation on Temporary IPv6 Interface Identifiers
draft-gont-6man-non-stable-iids-04

Abstract

This document specifies a set of requirements for generating temporary addresses, and clarifies the stability requirements for IPv6 addresses, allowing for the use of only temporary addresses.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

IPv6 StateLess Address AutoConfiguration (SLAAC) [RFC4862] has traditionally resulted in stable addresses, since the Interface Identifier (IID) has been generated by embedding a stable layer-2 numeric identifier (e.g., a MAC address). [RFC4941] originally implied, throughout the specification, that temporary addresses are generated and employed along with stable addresses.

While the use of stable addresses (only) or mixed stable and temporary addresses can be desirable in a number of scenarios, there are other scenarios in which, for security and privacy reasons, a node may want to use only temporary address (e.g., a temporary address).

On the other hand, the lack of a formal set of requirements for temporary addresses led to a number of flaws in popular implementations and in the protocol specification itself, such as allowing for the correlation of network activity carried out with different addresses, reusing randomized identifiers across different networks, etc.

This document clarifies the requirements for stability of IPv6 addresses, such that nodes are not required to configure stable addresses, and may instead employ only temporary addresses. It also specifies a set of requirements for the generation of temporary addresses.

2. Terminology

Statistically different:
When two values are required to be "statistically different", it means that the equality of those values cannot be caused by anything else other than random chance.

This document employs the definitions of "stable address" and "temporary address" from [RFC7721].

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 RFC 2119 [RFC2119].

3. Problem statement

When [RFC4941] was written, its authors wanted to prevent privacy and security attacks enabled by addresses that contain "an embedded interface identifier, which remains constant over time". They observed that "Anytime a fixed identifier is used in multiple contexts, it becomes possible to correlate seemingly unrelated activity using this identifier." They were concerned with both on-path attackers who would observe the IP addresses of packets observed in transit, and attackers that would have access to the logs of servers.

Since the publication of [RFC4941] in September 2007, our understanding of threats and mitigations has evolved. The IETF is now officially concerned with Pervasive Monitoring [RFC7258], as well as the wide spread collection of information for advertising and other purposes, for example through the Real Time Bidding protocol used for advertising auctions [RTB25].

3.1. Privacy requirements

The widespread deployment of encryption advocated in [RFC7624] is a response to Pervasive Monitoring. Encryption of communication reduces the amount of information that can be collected by monitoring data links, but does not prevent monitoring of IPv6 addresses embedded in clear text packet headers. Stable IPv6 addresses enable the correlation of such data over time.

MAC Address Randomization [IETFMACRandom] is another response to pervasive monitoring. In conjunction with DHCP Anonymity [RFC7844], it ensures that devices cannot be tracked by their MAC Address or their DHCP identifiers when they connect to "hot spots". However, the privacy effects of MAC Address Randomization would be nullified if a device kept using the same IPv6 address before and after a MAC-address randomization event.

Many Web Browsers have options enabling browsing "in private". However, if the web connections during the private mode use the same IPv6 address as those in the public mode, web tracking systems similar to [RTB25] will quickly find the correlation between the public personna of the user and the supposedly private connection. Similarly, many web browsers have options to "delete history", including deleting "cookies" and other persistent data. Again, if the same IPv6 address is used before and after the deletion of cookies, web tracking systems will easily correlate the new activity with the prior data collection.

Using temporary address alone may not be sufficient to prevent all forms of tracking. It is however quite clear that some usage of temporary addresses is necessary to provide user privacy. It is also clear that the usage of temporary addresses needs to be synchronized with other privacy defining event such as moving to a new network, performing MAC Address Randomization, or changing the privacy posture of a node.

4. Stability Requirements for IPv6 Addresses

Nodes are not required to generate addresses with any specific stability properties. That is, the generation of stable addresses is OPTIONAL. This means that a node may end up configuring only stable addresses, only temporary, or both stable and temporary addresses.

5. Requirements for Temporary IPv6 Addresses

The requirements for temporary IPv6 addresses are as follows:

1. Temporary addresses MUST have a limited lifetime (limited "valid lifetime" and "preferred lifetime" from [RFC4862]), that should be statistically different for different addresses. The lifetime of an address essentially limits the extent to which network activity correlation can be performed for such address.

2. The lifetime of an address MUST be further reduced when privacy-meaningful events (such as a node attaching to a new network) takes place.

3. The resulting Interface Identifiers MUST be statistically different when addresses are configured for different prefixes. That is, when temporary addresses are generated for different autoconfiguration prefixes for the same network interface, the resulting Interface Identifiers must be statistically different.

This means that, given two addresses that employ different prefixes, it must be difficult for an outside entity to tell whether the addresses correspond to the same network interface or even whether they have been generated by the same host.

4. It must be difficult for an outside entity to predict the Interface Identifiers that will be employed for temporary addresses, even with knowledge of the algorithm/method employed to generate them and/or knowledge of the Interface Identifiers previously employed for other temporary addresses.

5. The resulting Interface Identifiers MUST be semantically opaque [RFC7136] and MUST NOT follow any specific patterns.

By definition, temporary addresses have a limited lifetime. This is in contrast with e.g. stable addresses [RFC7217], that are not expected to become invalid under normal circumstances. Employing statistically different lifetimes for different addresses prevents an observer from synchronizing with the temporary address regeneration; that is, from being able to predict when a temporary address will become invalid and a new one regenerated, and thus being able to infer that one newly observed address is actually the result of regenerating a previously observed one.

The lifetime of an address should be further reduced by privacy-meaningful events. For example, a host must not employ the same address across network attachment events. That is, a host that de-attaches from a network and subsequently re-attaches to a (possibly different) network should regenerate all of its temporary addresses. Similarly, a host that implements MAC address randomization should regenerate all of its temporary addresses. Failure to regenerate temporary addresses upon such events would allow the correlation of network activity across such events (e.g., correlation of network activity as a host moves from one network to another). Other events, such as those discussed in Section 3.1 should also trigger the regeneration of all temporary addresses.

Temporary addresses configured for different prefixes should employ statistically different interface identifiers. In general, the reuse of identifiers across different contexts or scopes can be detrimental for security and privacy [I-D.gont-predictable-numeric-ids] [RFC6973] [RFC4941]. For example, a node that deterministically employs the same interface identifier for generating temporary addresses for different prefixes will allow the correlation of network activity.

For security and privacy reasons, the IIDs generated for temporary addresses must be unpredictable by an outside entity. Otherwise, the node may be subject to many (if not all) of the security and privacy issues that temporary addresses are expected to mitigate (please see [RFC7721]).

Any semantics or patterns in an IID might be leveraged by an attacker to e.g. reduce the search space when performing address-scanning attacks (see [RFC7707], infer the identity of the node, etc.

NOTE:
In the above text, where the "lifetime" of different addresses is required to be statistically different, or where the interface identifiers for different temporary addresses is required to be statistically different, the goal is that an implementation must not deterministically employ the same such values for different addresses. For example, where interface identifiers for different temporary addresses are required to be statistically different, the goal is to e.g. prevent an implementation from computing a single random interface identifier and employing such identifier for the generation of temporary addresses for other prefixes for the same network interface (as was the case with the algorithm specified in [RFC4941]). Therefore, a node is neither required nor expected to e.g. enforce that a newly-generated random interface identifier is not currently employed by any other temporary address configured by the node, or that such interface identifier has not been previously employed for any other temporary address configured by the node.

6. Future Work

This document clarifies the requirements for stability requirements for IPv6 addresses, and specifies requirements for temporary addresses. A separate document ([I-D.gont-taps-address-usage-problem-statement]) discusses the trade-offs involved when considering different stability properties of IPv6 addresses.

7. IANA Considerations

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

8. Security Considerations

This document clarifies the stability requirements for IPv6 addresses, and specifies requirements for the generation of temporary addresses.

The security and privacy properties of IPv6 addresses have been discussed in detail in [RFC7721] and [RFC7707].

9. Acknowledgements

The authors would like to thank (in alphabetical order) Brian Carpenter, Lorenzo Colitti, and David Plonka, for providing valuable feedback on earlier versions of this document.

10. References

10.1. Normative References

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

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

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

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <https://www.rfc-editor.org/info/rfc4941>.

[RFC5453] Krishnan, S., "Reserved IPv6 Interface Identifiers", RFC 5453, DOI 10.17487/RFC5453, February 2009, <https://www.rfc-editor.org/info/rfc5453>.

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

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

[RFC8064] Gont, F., Cooper, A., Thaler, D., and W. Liu, "Recommendation on Stable IPv6 Interface Identifiers", RFC 8064, DOI 10.17487/RFC8064, February 2017, <https://www.rfc-editor.org/info/rfc8064>.

10.2. Informative References

[FIPS-SHS] NIST, "Secure Hash Standard (SHS)", FIPS Publication 180-4, March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[I-D.gont-predictable-numeric-ids] Gont, F. and I. Arce, "Security and Privacy Implications of Numeric Identifiers Employed in Network Protocols", draft-gont-predictable-numeric-ids-02 (work in progress), February 2018.

[I-D.gont-taps-address-usage-problem-statement] Gont, F., Gont, G., Corbo, M., and C. Huitema, "Problem Statement Regarding IPv6 Address Usage", draft-gont-taps-address-usage-problem-statement-00 (work in progress), February 2018.

[IANA-RESERVED-IID] IANA, "Reserved IPv6 Interface Identifiers", <http://www.iana.org/assignments/ipv6-interface-ids>.

[IETFMACRandom] Zuniga, JC., "MAC Privacy", November 2014, <http://www.ietf.org/blog/2014/11/mac-privacy/>.

[OPEN-GROUP] The Open Group, "The Open Group Base Specifications Issue 7 / IEEE Std 1003.1-2008, 2016 Edition", Section 4.16 Seconds Since the Epoch, 2016, <http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/contents.html>.

[RAID2015]
Ullrich, J. and E. Weippl, "Privacy is Not an Option: Attacking the IPv6 Privacy Extension", International Symposium on Recent Advances in Intrusion Detection (RAID), 2015, <https://www.sba-research.org/wp-content/uploads/publications/Ullrich2015Privacy.pdf>.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC3041] Narten, T. and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 3041, DOI 10.17487/RFC3041, January 2001, <https://www.rfc-editor.org/info/rfc3041>.

[RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for Detecting Network Attachment in IPv6", RFC 6059, DOI 10.17487/RFC6059, November 2010, <https://www.rfc-editor.org/info/rfc6059>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[RFC7624] Barnes, R., Schneier, B., Jennings, C., Hardie, T., Trammell, B., Huitema, C., and D. Borkmann, "Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement", RFC 7624, DOI 10.17487/RFC7624, August 2015, <https://www.rfc-editor.org/info/rfc7624>.

[RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", RFC 7707, DOI 10.17487/RFC7707, March 2016, <https://www.rfc-editor.org/info/rfc7707>.

[RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <https://www.rfc-editor.org/info/rfc7721>.

[RFC7844] Huitema, C., Mrugalski, T., and S. Krishnan, "Anonymity Profiles for DHCP Clients", RFC 7844, DOI 10.17487/RFC7844, May 2016, <https://www.rfc-editor.org/info/rfc7844>.

[RTB25] Interactive Advertising Bureau (IAB), "Real Time Bidding (RTB) project, OpenRTB API Specification Version 2.5", December 2016, <http://www.iab.com/wp-content/uploads/2016/03/OpenRTB-API-Specification-Version-2-5-FINAL.pdf>.

Authors' Addresses

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

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

Christian Huitema
Private Octopus Inc.
Friday Harbor, WA 98250
U.S.A.

Email: huitema@huitema.net
URI: http://privateoctopus.com/

Suresh Krishnan
Ericsson Research
8400 Decarie Blvd.
Town of Mount Royal, QC
Canada

Email: suresh.krishnan@ericsson.com

Guillermo Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: ggont@si6networks.com
URI: https://www.si6networks.com

Madeleine Garcia Corbo
Servicios de Informacion del Transporte
Neptuno 358
Havana City 10400
Cuba

Email: madelen.garcia16@gmail.com

IPv6 maintenance Working Group (6man)
Internet-Draft
Updates: 2464, 2467, 2470, 2491, 2492,
2497, 2590, 3146, 3572, 4291,
4338, 4391, 5072, 5121 (if
approved)
Intended status: Standards Track
Expires: March 29, 2017
F. Gont
SI6 Networks / UTN-FRH
A. Cooper
Cisco
D. Thaler
Microsoft
W. Liu
Huawei Technologies
September 28, 2016

Recommendation on Stable IPv6 Interface Identifiers
draft-ietf-6man-default-iids-16

Abstract

This document changes the recommended default IID generation scheme for cases where SLAAC is used to generate a stable IPv6 address. It recommends using the mechanism specified in RFC7217 in such cases, and recommends against embedding stable link-layer addresses in IPv6 Interface Identifiers. It formally updates RFC2464, RFC2467, RFC2470, RFC2491, RFC2492, RFC2497, RFC2590, RFC3146, RFC3572, RFC4291, RFC4338, RFC4391, RFC5072, and RFC5121. This document does not change any existing recommendations concerning the use of temporary addresses as specified in RFC 4941.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[RFC4862] specifies Stateless Address Autoconfiguration (SLAAC) for IPv6 [RFC2460], which typically results in hosts configuring one or more "stable" addresses composed of a network prefix advertised by a local router, and an Interface Identifier (IID) [RFC4291] that typically embeds a stable link-layer address (e.g., an IEEE LAN MAC address).

In some network technologies and adaptation layers, the use of an IID based on a link-layer address may offer some advantages. For example, the IP-over-IEEE802.15.4 standard in [RFC6775] allows for compression of IPv6 addresses when the IID is based on the underlying link-layer address.

The security and privacy implications of embedding a stable link-layer address in an IPv6 IID have been known for some time now, and are discussed in great detail in [RFC7721]. They include:

- Network activity correlation
- Location tracking
- Address scanning
- Device-specific vulnerability exploitation

More generally, the reuse of identifiers that have their own semantics or properties across different contexts or scopes can be detrimental for security and privacy [I-D.gont-predictable-numeric-ids]. In the case of traditional stable IPv6 IIDs, some of the security and privacy implications are dependent on the properties of the underlying link-layer addresses (e.g., whether the link-layer address is ephemeral or randomly generated), while other implications (e.g., reduction of the entropy of the IID) depend on the algorithm for generating the IID itself. In standardized recommendations for stable IPv6 IID generation meant to achieve particular security and privacy properties, it is therefore necessary to recommend against embedding stable link-layer addresses in IPv6 IIDs.

Furthermore, some popular IPv6 implementations have already deviated from the traditional stable IID generation scheme to mitigate the aforementioned security and privacy implications [Microsoft].

As a result of the aforementioned issues, this document changes the recommended default IID generation scheme for generating stable IPv6 addresses with SLAAC to that specified in [RFC7217], and recommends against embedding stable link-layer addresses in IPv6 Interface Identifiers, such that the aforementioned issues are mitigated. That is, this document simply replaces the default algorithm that is recommended to be employed when generating stable IPv6 IIDs.

NOTE: [RFC4291] defines the "Modified EUI-64 format" for IIDs. Appendix A of [RFC4291] then describes how to transform an IEEE EUI-64 identifier, or an IEEE 802 48-bit MAC address from which an EUI-64 identifier is derived, into an IID in the Modified EUI-64 format.

In a variety of scenarios, addresses that remain stable for the lifetime of a host’s connection to a single subnet, are viewed as desirable. For example, stable addresses may be viewed as beneficial for network management, event logging, enforcement of access control, provision of quality of service, or for server or routing interfaces. Similarly, stable addresses (as opposed to temporary addresses [RFC4941]) allow for long-lived TCP connections, and are also usually desirable when performing server-like functions (i.e., receiving incoming connections).

The recommendations in this document apply only in cases where implementations otherwise would have configured a stable IPv6 IID containing a link layer address. For example, this document does not change any existing recommendations concerning the use of temporary addresses as specified in [RFC4941], nor do the recommendations apply to cases where SLAAC is employed to generate non-stable IPv6 addresses (e.g. by embedding a link-layer address that is periodically randomized), nor does it introduce any new requirements regarding when stable addresses are to be configured. Thus, the recommendations in this document simply improve the security and privacy properties of stable addresses.

2. Terminology

Stable address:
An address that does not vary over time within the same network (as defined in [RFC7721]).

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 RFC 2119 [RFC2119].

3. Generation of IPv6 Interface Identifiers with SLAAC

Nodes SHOULD implement and employ [RFC7217] as the default scheme for generating stable IPv6 addresses with SLAAC. A link layer MAY also define a mechanism for stable IPv6 address generation that is more efficient and does not address the security and privacy considerations discussed in Section 1. The choice of whether to enable the security- and privacy-preserving mechanism or not SHOULD be configurable in such a case.

By default, nodes SHOULD NOT employ IPv6 address generation schemes that embed a stable link-layer address in the IID. In particular, this document RECOMMENDS that nodes do not generate stable IIDs with the schemes specified in [RFC2464], [RFC2467], [RFC2470], [RFC2491], [RFC2492], [RFC2497], [RFC2590], [RFC3146], [RFC3572], [RFC4338], [RFC4391], [RFC5121], and [RFC5072].

4. Future Work

At the time of this writing, the mechanisms specified in the following documents might require updates to be fully compatible with the recommendations in this document:

- "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks" [RFC6282]

- "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944]

- "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)"[RFC6775] o "Transmission of IPv6 Packets over ITU-T G.9959 Networks" [RFC7428]

Future revisions or updates of these documents should take the issues of privacy and security mentioned in Section 1 and explain any design and engineering considerations that lead to the use of stable IIDs based on a node’s link-layer address.

5. 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.

6. Security Considerations

This recommends against the (default) use of predictable Interface Identifiers in IPv6 addresses. It recommends [RFC7217] as the default scheme for generating IPv6 stable addresses with SLAAC, such that the security and privacy issues of IIDs that embed stable link-layer addresses are mitigated.

7. Acknowledgements

The authors would like to thank (in alphabetical order) Bob Hinden, Ray Hunter and Erik Nordmark, for providing a detailed review of this document.

The authors would like to thank (in alphabetical order) Fred Baker, Carsten Bormann, Scott Brim, Brian Carpenter, Samita Chakrabarti, Tim Chown, Lorenzo Colitti, Jean-Michel Combes, Greg Daley, Esko Dijk, Ralph Droms, David Farmer, Brian Haberman, Ulrich Herberg, Philip Homburg, Jahangir Hossain, Jonathan Hui, Christian Huitema, Ray Hunter, Erik Kline, Sheng Jiang, Roger Jorgensen, Dan Luedtke, Kerry Lynn, George Mitchel, Gabriel Montenegro, Erik Nordmark, Simon Perreault, Tom Petch, Alexandru Petrescu, Michael Richardson, Arturo Servin, Mark Smith, Tom Taylor, Ole Troan, Tina Tsou, Glen Turner, Randy Turner, James Woodyatt, and Juan Carlos Zuniga, for providing valuable comments on earlier versions of this document.

8. References

8.1. Normative References

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

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

[RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <http://www.rfc-editor.org/info/rfc2464>.

[RFC2467] Crawford, M., "Transmission of IPv6 Packets over FDDI Networks", RFC 2467, DOI 10.17487/RFC2467, December 1998, <http://www.rfc-editor.org/info/rfc2467>.

[RFC2470] Crawford, M., Narten, T., and S. Thomas, "Transmission of IPv6 Packets over Token Ring Networks", RFC 2470, DOI 10.17487/RFC2470, December 1998, <http://www.rfc-editor.org/info/rfc2470>.

[RFC2491] Armitage, G., Schulter, P., Jork, M., and G. Harter, "IPv6 over Non-Broadcast Multiple Access (NBMA) networks", RFC 2491, DOI 10.17487/RFC2491, January 1999, <http://www.rfc-editor.org/info/rfc2491>.

[RFC2492] Armitage, G., Schulter, P., and M. Jork, "IPv6 over ATM Networks", RFC 2492, DOI 10.17487/RFC2492, January 1999, <http://www.rfc-editor.org/info/rfc2492>.

[RFC2497] Souvatzis, I., "Transmission of IPv6 Packets over ARCnet Networks", RFC 2497, DOI 10.17487/RFC2497, January 1999, <http://www.rfc-editor.org/info/rfc2497>.

[RFC2590] Conta, A., Malis, A., and M. Mueller, "Transmission of IPv6 Packets over Frame Relay Networks Specification", RFC 2590, DOI 10.17487/RFC2590, May 1999, <http://www.rfc-editor.org/info/rfc2590>.

[RFC3146] Fujisawa, K. and A. Onoe, "Transmission of IPv6 Packets over IEEE 1394 Networks", RFC 3146, DOI 10.17487/RFC3146, October 2001, <http://www.rfc-editor.org/info/rfc3146>.

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

[RFC4338] DeSanti, C., Carlson, C., and R. Nixon, "Transmission of IPv6, IPv4, and Address Resolution Protocol (ARP) Packets over Fibre Channel", RFC 4338, DOI 10.17487/RFC4338, January 2006, <http://www.rfc-editor.org/info/rfc4338>.

[RFC4391] Chu, J. and V. Kashyap, "Transmission of IP over InfiniBand (IPoIB)", RFC 4391, DOI 10.17487/RFC4391, April 2006, <http://www.rfc-editor.org/info/rfc4391>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <http://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <http://www.rfc-editor.org/info/rfc4944>.

[RFC5072] Varada, S., Ed., Haskins, D., and E. Allen, "IP Version 6 over PPP", RFC 5072, DOI 10.17487/RFC5072, September 2007, <http://www.rfc-editor.org/info/rfc5072>.

[RFC5121] Patil, B., Xia, F., Sarikaya, B., Choi, JH., and S. Madanapalli, "Transmission of IPv6 via the IPv6 Convergence Sublayer over IEEE 802.16 Networks", RFC 5121, DOI 10.17487/RFC5121, February 2008, <http://www.rfc-editor.org/info/rfc5121>.

[RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <http://www.rfc-editor.org/info/rfc6282>.

[RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <http://www.rfc-editor.org/info/rfc6775>.

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

[RFC7428] Brandt, A. and J. Buron, "Transmission of IPv6 Packets over ITU-T G.9959 Networks", RFC 7428, DOI 10.17487/RFC7428, February 2015, <http://www.rfc-editor.org/info/rfc7428>.

8.2. Informative References

[I-D.gont-predictable-numeric-ids] Gont, F. and I. Arce, "Security and Privacy Implications of Numeric Identifiers Employed in Network Protocols", draft-gont-predictable-numeric-ids-00 (work in progress), February 2016.

[Microsoft] Davies, J., "Understanding IPv6, 3rd. ed", page 83, Microsoft Press, 2012, <http://it-ebooks.info/book/1022/>.

[RFC3572] Ogura, T., Maruyama, M., and T. Yoshida, "Internet Protocol Version 6 over MAPOS (Multiple Access Protocol Over SONET/SDH)", RFC 3572, DOI 10.17487/RFC3572, July 2003, <http://www.rfc-editor.org/info/rfc3572>.

[RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <http://www.rfc-editor.org/info/rfc7721>.

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

Alissa Cooper
Cisco
707 Tasman Drive
Milpitas, CA 95035
US

Phone: +1-408-902-3950
Email: alcoop@cisco.com
URI: https://www.cisco.com/

Dave Thaler
Microsoft
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Phone: +1 425 703 8835
Email: dthaler@microsoft.com

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

Email: liushucheng@huawei.com

Network Working Group J. McCann
Internet-Draft Digital Equipment Corporation
Obsoletes: 1981 (if approved) S. Deering
Intended status: Standards Track Retired
Expires: November 28, 2017 J. Mogul
Digital Equipment Corporation
R. Hinden, Ed.
Check Point Software
May 27, 2017

Path MTU Discovery for IP version 6
draft-ietf-6man-rfc1981bis-08

Abstract

This document describes Path MTU Discovery for IP version 6. It is largely derived from RFC 1191, which describes Path MTU Discovery for IP version 4. It obsoletes RFC1981.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

When one IPv6 node has a large amount of data to send to another node, the data is transmitted in a series of IPv6 packets. These packets can have a size less than or equal to the Path MTU (PMTU). Alternatively, they can be larger packets that are fragmented into a series of fragments each with a size less than or equal to the PMTU.

It is usually preferable that these packets be of the largest size that can successfully traverse the path from the source node to the destination node without the need for IPv6 fragmentation. This packet size is referred to as the Path MTU, and it is equal to the minimum link MTU of all the links in a path. This document defines a standard mechanism for a node to discover the PMTU of an arbitrary path.

IPv6 nodes should implement Path MTU Discovery in order to discover and take advantage of paths with PMTU greater than the IPv6 minimum link MTU [I-D.ietf-6man-rfc2460bis]. A minimal IPv6 implementation (e.g., in a boot ROM) may choose to omit implementation of Path MTU Discovery.

Nodes not implementing Path MTU Discovery must use the IPv6 minimum link MTU defined in [I-D.ietf-6man-rfc2460bis] as the maximum packet size. In most cases, this will result in the use of smaller packets than necessary, because most paths have a PMTU greater than the IPv6 minimum link MTU. A node sending packets much smaller than the Path MTU allows is wasting network resources and probably getting suboptimal throughput.

Nodes implementing Path MTU Discovery and sending packets larger than the IPv6 minimum link MTU are susceptible to problematic connectivity if ICMPv6 [ICMPv6] messages are blocked or not transmitted. For example, this will result in connections that complete the TCP three-way handshake correctly but then hang when data is transferred. This state is referred to as a black hole connection [RFC2923]. Path MTU Discovery relies on ICMPv6 Packet Too Big (PTB) to determine the MTU of the path.

An extension to Path MTU Discovery defined in this document can be found in [RFC4821]. RFC4821 defines a method for Packetization Layer Path MTU Discovery (PLPMTUD) designed for use over paths where delivery of ICMPv6 messages to a host is not assured.

Note: This document is an update to [RFC1981] that was published prior to [RFC2119] being published. Consequently although RFC1981 used the "should/must" style language in upper and lower case, this document does not cite the RFC2119 definitions and only uses lower case for these words.

2. Terminology

node — a device that implements IPv6.

router — a node that forwards IPv6 packets not explicitly addressed to itself.

host	any node that is not a router.
upper layer	a protocol layer immediately above IPv6. Examples are transport protocols such as TCP and UDP, control protocols such as ICMPv6, routing protocols such as OSPF, and internet or lower-layer protocols being "tunneled" over (i.e., encapsulated in) IPv6 such as IPX, AppleTalk, or IPv6 itself.
link	a communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IPv6. Examples are Ethernets (simple or bridged); PPP links; X.25, Frame Relay, or ATM networks; and internet (or higher) layer "tunnels", such as tunnels over IPv4 or IPv6 itself.
interface	a node’s attachment to a link.
address	an IPv6-layer identifier for an interface or a set of interfaces.
packet	an IPv6 header plus payload. The packet can have a size less than or equal to the PMTU. Alternatively, this can be a larger packet that is fragmented into a series of fragments each with a size less than or equal to the PMTU.
link MTU	the maximum transmission unit, i.e., maximum packet size in octets, that can be conveyed in one piece over a link.
path	the set of links traversed by a packet between a source node and a destination node.
path MTU	the minimum link MTU of all the links in a path between a source node and a destination node.
PMTU	path MTU
Path MTU Discovery	process by which a node learns the PMTU of a path
EMTU_S	Effective MTU for sending, used by upper layer protocols to limit the size of IP packets they queue for sending [RFC6691] [RFC1122].

EMTU_R — Effective MTU for receiving, the largest packet that can be reassembled at the receiver [RFC1122].

flow — a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers.

flow id — a combination of a source address and a non-zero flow label.

3. Protocol Overview

This memo describes a technique to dynamically discover the PMTU of a path. The basic idea is that a source node initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path. If any of the packets sent on that path are too large to be forwarded by some node along the path, that node will discard them and return ICMPv6 Packet Too Big messages. Upon receipt of such a message, the source node reduces its assumed PMTU for the path based on the MTU of the constricting hop as reported in the Packet Too Big message. The decreased PMTU causes the source to send smaller packets or change EMTU_S to cause upper layer to reduce the size of IP packets it sends.

The Path MTU Discovery process ends when the source node's estimate of the PMTU is less than or equal to the actual PMTU. Note that several iterations of the packet-sent/Packet-Too-Big-message-received cycle may occur before the Path MTU Discovery process ends, as there may be links with smaller MTUs further along the path.

Alternatively, the node may elect to end the discovery process by ceasing to send packets larger than the IPv6 minimum link MTU.

The PMTU of a path may change over time, due to changes in the routing topology. Reductions of the PMTU are detected by Packet Too Big messages. To detect increases in a path's PMTU, a node periodically increases its assumed PMTU. This will almost always result in packets being discarded and Packet Too Big messages being generated, because in most cases the PMTU of the path will not have changed. Therefore, attempts to detect increases in a path's PMTU should be done infrequently.

Path MTU Discovery supports multicast as well as unicast destinations. In the case of a multicast destination, copies of a packet may traverse many different paths to many different nodes. Each path may have a different PMTU, and a single multicast packet may result in multiple Packet Too Big messages, each reporting a different next-hop MTU. The minimum PMTU value across the set of paths in use determines the size of subsequent packets sent to the multicast destination.

Note that Path MTU Discovery must be performed even in cases where a node "thinks" a destination is attached to the same link as itself, it might have a PMTU lower than the link MTU. In a situation such as when a neighboring router acts as proxy [ND] for some destination, the destination can appear to be directly connected but it is in fact more than one hop away.

4. Protocol Requirements

As discussed in Section 1, IPv6 nodes are not required to implement Path MTU Discovery. The requirements in this section apply only to those implementations that include Path MTU Discovery.

Nodes should appropriately validate the payload of ICMPv6 PTB messages to ensure these are received in response to transmitted traffic (i.e., a reported error condition that corresponds to an IPv6 packet actually sent by the application) per [ICMPv6].

If a node receives a Packet Too Big message reporting a next-hop MTU that is less than the IPv6 minimum link MTU, it must discard it. A node must not reduce its estimate of the Path MTU below the IPv6 minimum link MTU on receipt of an Packet Too Big message.

When a node receives a Packet Too Big message, it must reduce its estimate of the PMTU for the relevant path, based on the value of the MTU field in the message. The precise behavior of a node in this circumstance is not specified, since different applications may have different requirements, and since different implementation architectures may favor different strategies.

After receiving a Packet Too Big message, a node must attempt to avoid eliciting more such messages in the near future. The node must reduce the size of the packets it is sending along the path. Using a PMTU estimate larger than the IPv6 minimum link MTU may continue to elicit Packet Too Big messages. Because each of these messages (and the dropped packets they respond to) consume network resources, Nodes using Path MTU Discovery must detect decreases in PMTU as fast as possible.

Nodes may detect increases in PMTU, but because doing so requires sending packets larger than the current estimated PMTU, and because the likelihood is that the PMTU will not have increased, this must be done at infrequent intervals. An attempt to detect an increase (by sending a packet larger than the current estimate) must not be done less than 5 minutes after a Packet Too Big message has been received for the given path. The recommended setting for this timer is twice its minimum value (10 minutes).

A node must not increase its estimate of the Path MTU in response to the contents of a Packet Too Big message. A message purporting to announce an increase in the Path MTU might be a stale packet that has been floating around in the network, a false packet injected as part of a denial-of-service attack, or the result of having multiple paths to the destination, each with a different PMTU.

5. Implementation Issues

This section discusses a number of issues related to the implementation of Path MTU Discovery. This is not a specification, but rather a set of notes provided as an aid for implementers.

The issues include:

- What layer or layers implement Path MTU Discovery?

- How is the PMTU information cached?

- How is stale PMTU information removed?

- What must transport and higher layers do?

5.1. Layering

In the IP architecture, the choice of what size packet to send is made by a protocol at a layer above IP. This memo refers to such a protocol as a "packetization protocol". Packetization protocols are usually transport protocols (for example, TCP) but can also be higher-layer protocols (for example, protocols built on top of UDP).

Implementing Path MTU Discovery in the packetization layers simplifies some of the inter-layer issues, but has several drawbacks: the implementation may have to be redone for each packetization protocol, it becomes hard to share PMTU information between different packetization layers, and the connection-oriented state maintained by some packetization layers may not easily extend to save PMTU information for long periods.

It is therefore suggested that the IP layer store PMTU information and that the ICMPv6 layer process received Packet Too Big messages. The packetization layers may respond to changes in the PMTU by changing the size of the messages they send. To support this layering, packetization layers require a way to learn of changes in the value of MMS_S, the "maximum send transport-message size" [RFC1122].

MMS_S is a transport message size calculated by subtracting the size of the IPv6 header (including IPv6 extension headers) from the largest IP packet that can be sent, EMTU_S. MMS_S is limited by a combination of factors, including the PMTU, support for packet fragmentation and reassembly, and the packet reassembly limit (see [I-D.ietf-6man-rfc2460bis] section "Fragment Header"). When source fragmentation is available, EMTU_S is set to EMTU_R, as indicated by the receiver using an upper layer protocol or based on protocol requirements (1500 octets for IPv6). When a message larger than PMTU is to be transmitted, the source creates fragments, each limited by PMTU. When source fragmentation is not desired, EMTU_S is set to PMTU, and the upper layer protocol is expected to either perform its own fragmentation and reassembly or otherwise limit the size of its messages accordingly.

However, packetization layers are encouraged to avoid sending messages that will require source fragmentation (for the case against fragmentation, see [FRAG]).

5.2. Storing PMTU information

Ideally, a PMTU value should be associated with a specific path traversed by packets exchanged between the source and destination nodes. However, in most cases a node will not have enough information to completely and accurately identify such a path. Rather, a node must associate a PMTU value with some local representation of a path. It is left to the implementation to select the local representation of a path. For nodes with multiple interfaces, Path MTU information should be maintained for each IPv6 link.

In the case of a multicast destination address, copies of a packet may traverse many different paths to reach many different nodes. The local representation of the "path" to a multicast destination must represent a potentially large set of paths.

Minimally, an implementation could maintain a single PMTU value to be used for all packets originated from the node. This PMTU value would be the minimum PMTU learned across the set of all paths in use by the node. This approach is likely to result in the use of smaller packets than is necessary for many paths. In the case of multipath routing (e.g., Equal Cost Multipath Routing (ECMP)), a set of paths can exist even for a single source and destination pair.

An implementation could use the destination address as the local representation of a path. The PMTU value associated with a destination would be the minimum PMTU learned across the set of all paths in use to that destination. This approach will result in the use of optimally sized packets on a per-destination basis. This approach integrates nicely with the conceptual model of a host as described in [ND]: a PMTU value could be stored with the corresponding entry in the destination cache.

If flows [I-D.ietf-6man-rfc2460bis] are in use, an implementation could use the flow id as the local representation of a path. Packets sent to a particular destination but belonging to different flows may use different paths, as with ECMP, in which the choice of path might depending on the flow id. This approach might result in the use of optimally sized packets on a per-flow basis, providing finer granularity than PMTU values maintained on a per-destination basis.

For source routed packets (i.e. packets containing an IPv6 Routing header [I-D.ietf-6man-rfc2460bis]), the source route may further qualify the local representation of a path.

Initially, the PMTU value for a path is assumed to be the (known) MTU of the first-hop link.

When a Packet Too Big message is received, the node determines which path the message applies to based on the contents of the Packet Too Big message. For example, if the destination address is used as the local representation of a path, the destination address from the original packet would be used to determine which path the message applies to.

> Note: if the original packet contained a Routing header, the Routing header should be used to determine the location of the destination address within the original packet. If Segments Left is equal to zero, the destination address is in the Destination Address field in the IPv6 header. If Segments Left is greater than zero, the destination address is the last address (Address[n]) in the Routing header.

The node then uses the value in the MTU field in the Packet Too Big message as a tentative PMTU value or the IPv6 minimum link MTU if that is larger, and compares the tentative PMTU to the existing PMTU. If the tentative PMTU is less than the existing PMTU estimate, the tentative PMTU replaces the existing PMTU as the PMTU value for the path.

The packetization layers must be notified about decreases in the PMTU. Any packetization layer instance (for example, a TCP connection) that is actively using the path must be notified if the PMTU estimate is decreased.

Note: even if the Packet Too Big message contains an Original Packet Header that refers to a UDP packet, the TCP layer must be notified if any of its connections use the given path.

Also, the instance that sent the packet that elicited the Packet Too Big message should be notified that its packet has been dropped, even if the PMTU estimate has not changed, so that it may retransmit the dropped data.

Note: An implementation can avoid the use of an asynchronous notification mechanism for PMTU decreases by postponing notification until the next attempt to send a packet larger than the PMTU estimate. In this approach, when an attempt is made to SEND a packet that is larger than the PMTU estimate, the SEND function should fail and return a suitable error indication. This approach may be more suitable to a connectionless packetization layer (such as one using UDP), which (in some implementations) may be hard to "notify" from the ICMPv6 layer. In this case, the normal timeout-based retransmission mechanisms would be used to recover from the dropped packets.

It is important to understand that the notification of the packetization layer instances using the path about the change in the PMTU is distinct from the notification of a specific instance that a packet has been dropped. The latter should be done as soon as practical (i.e., asynchronously from the point of view of the packetization layer instance), while the former may be delayed until a packetization layer instance wants to create a packet.

5.3. Purging stale PMTU information

Internetwork topology is dynamic; routes change over time. While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale.

If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values. When a PMTU value has not been decreased for a while (on the order of 10 minutes), it should probe to find if a larger PMTU is supported.

Note: an implementation should provide a means for changing the timeout duration, including setting it to "infinity". For example, nodes attached to a link with a large MTU which is then attached to the rest of the Internet via a link with a small MTU are never going to discover a new non-local PMTU, so they should not have to put up with dropped packets every 10 minutes.

5.4. Packetization layer actions

A packetization layer (e.g., TCP) must use the PMTU for the path(s) in use by a connection; it should not send segments that would result in packets larger than the PMTU, except to probe during PMTU discovery (this probe packet must not be fragmented to the PMTU). A simple implementation could ask the IP layer for this value each time it created a new segment, but this could be inefficient. An implementation typically caches other values derived from the PMTU. It may be simpler to receive asynchronous notification when the PMTU changes, so that these variables may be also updated.

A TCP implementation must also store the Maximum Segment Size (MSS) value received from its peer, which represents the EMTU_R, the largest packet that can be reassembled by the receiver, and must not send any segment larger than this MSS, regardless of the PMTU.

The value sent in the TCP MSS option is independent of the PMTU; it is determined by the receiver reassembly limit EMTU_R. This MSS option value is used by the other end of the connection, which may be using an unrelated PMTU value. See [I-D.ietf-6man-rfc2460bis] sections "Packet Size Issues" and "Maximum Upper-Layer Payload Size" for information on selecting a value for the TCP MSS option.

Reception of a Packet Too Big message implies that a packet was dropped by the node that sent the ICMPv6 message. A reliable upper layer protocol will detect this loss by its own means, and recover it by its normal retransmission methods. The retransmission could result in delay, depending on the loss detection method used by the upper layer protocol. If the Path MTU Discovery process requires several steps to find the PMTU of the full path, this could finally delay the retransmission by many round-trip times.

Alternatively, the retransmission could be done in immediate response to a notification that the Path MTU was decreased, but only for the specific connection specified by the Packet Too Big message, but only based on the message and connection. The packet size used in the retransmission should be no larger than the new PMTU.

> Note: A packetization layer that determines a probe packet is lost, needs to adapt the segment size of the retransmission. Using the reported size in the last Packet Too Big message, however, can lead to further losses as there might be smaller PMTU

limits at the routers further along the path. This would lead to loss of all retransmitted segments and therefore cause unnecessary congestion as well as additional packets to be sent each time a new router announces a smaller MTU. Any packetization layer that uses retransmission is therefore also responsible for congestion control of its retransmissions [RFC8085].

A loss caused by a PMTU probe indicated by the reception of a Packet Too Big message must not be considered as a congestion notification and hence the congestion window may not change.

5.5. Issues for other transport protocols

Some transport protocols are not allowed to repacketize when doing a retransmission. That is, once an attempt is made to transmit a segment of a certain size, the transport cannot split the contents of the segment into smaller segments for retransmission. In such a case, the original segment can be fragmented by the IP layer during retransmission. Subsequent segments, when transmitted for the first time, should be no larger than allowed by the Path MTU.

Path MTU Discovery for IPv4 [RFC1191] used NFS as an example of a UDP-based application that benefits from PMTU discovery. Since then [RFC7530], states the supported transport layer between NFS and IP must be an IETF standardized transport protocol that is specified to avoid network congestion; such transports include TCP, Stream Control Transmission Protocol (SCTP) [RFC4960], and the Datagram Congestion Control Protocol (DCCP) [RFC4340]. In this case, the transport is responsible for ensuring that transmitted segments (except probes) conform to the the Path MTU, including supporting PMTU discovery probe transmissions as needed.

5.6. Management interface

It is suggested that an implementation provide a way for a system utility program to:

- Specify that Path MTU Discovery not be done on a given path.

- Change the PMTU value associated with a given path.

The former can be accomplished by associating a flag with the path; when a packet is sent on a path with this flag set, the IP layer does not send packets larger than the IPv6 minimum link MTU.

These features might be used to work around an anomalous situation, or by a routing protocol implementation that is able to obtain Path MTU values.

The implementation should also provide a way to change the timeout period for aging stale PMTU information.

6. Security Considerations

This Path MTU Discovery mechanism makes possible two denial-of-service attacks, both based on a malicious party sending false Packet Too Big messages to a node.

In the first attack, the false message indicates a PMTU much smaller than reality. In response, the victim node should never set its PMTU estimate below the IPv6 minimum link MTU. A sender that falsely reduces to this MTU would observe suboptimal performance.

In the second attack, the false message indicates a PMTU larger than reality. If believed, this could cause temporary blockage as the victim sends packets that will be dropped by some router. Within one round-trip time, the node would discover its mistake (receiving Packet Too Big messages from that router), but frequent repetition of this attack could cause lots of packets to be dropped. A node, however, must not raise its estimate of the PMTU based on a Packet Too Big message, so should not be vulnerable to this attack.

Both of these attacks can cause a black hole connection, that is, the TCP three-way handshake completes correctly but the connection hangs when data is transfered.

A malicious party could also cause problems if it could stop a victim from receiving legitimate Packet Too Big messages, but in this case there are simpler denial-of-service attacks available.

If ICMPv6 filtering prevents reception of ICMPv6 Packet Too Big messages, the source will not learn the actual path MTU. Packetization Layer Path MTU Discovery [RFC4821] does not rely upon network support for ICMPv6 messages and is therefore considered more robust than standard PMTUD. It is not susceptible to "black holed" connections caused by filtering of ICMPv6 message. See [RFC4890] for recommendations regarding filtering ICMPv6 messages.

7. Acknowledgements

We would like to acknowledge the authors of and contributors to [RFC1191], from which the majority of this document was derived. We would also like to acknowledge the members of the IPng working group for their careful review and constructive criticisms.

We would also like to acknowledge the contributors to this update of "Path MTU Discovery for IP version 6". This includes members of the 6MAN w.g., area directorate reviewers, the IESG, and especially to Joe Touch and Gorry Fairhurst.

8. IANA Considerations

This document does not have any IANA actions

9. References

9.1. Normative References

[I-D.ietf-6man-rfc2460bis]
Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", draft-ietf-6man-rfc2460bis-13 (work in progress), May 2017.

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

9.2. Informative References

[FRAG] Kent, C. and J. Mogul, "Fragmentation Considered Harmful", In Proc. SIGCOMM ’87 Workshop on Frontiers in Computer Communications Technology , August 1987.

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

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

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

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <http://www.rfc-editor.org/info/rfc1981>.

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

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

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

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <http://www.rfc-editor.org/info/rfc4821>.

[RFC4890] Davies, E. and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls", RFC 4890, DOI 10.17487/RFC4890, May 2007, <http://www.rfc-editor.org/info/rfc4890>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

[RFC6691] Borman, D., "TCP Options and Maximum Segment Size (MSS)", RFC 6691, DOI 10.17487/RFC6691, July 2012, <http://www.rfc-editor.org/info/rfc6691>.

[RFC7530] Haynes, T., Ed. and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", RFC 7530, DOI 10.17487/RFC7530, March 2015, <http://www.rfc-editor.org/info/rfc7530>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <http://www.rfc-editor.org/info/rfc8085>.

Appendix A. Comparison to RFC 1191

This document is based in large part on RFC 1191, which describes Path MTU Discovery for IPv4. Certain portions of RFC 1191 were not needed in this document:

router specification Packet Too Big messages and corresponding router behavior are defined in [ICMPv6]

Don't Fragment bit	there is no DF bit in IPv6 packets
TCP MSS discussion	selecting a value to send in the TCP MSS option is discussed in [I-D.ietf-6man-rfc2460bis]
old-style messages	all Packet Too Big messages report the MTU of the constricting link
MTU plateau tables	not needed because there are no old-style messages

Appendix B. Changes Since RFC 1981

This document is based on RFC1981 has the following changes from RFC1981:

- o Clarified Section 1 "Introduction" that the purpose of PMTUD is to reduce the need for IPv6 fragmentation.

- o Added text to Section 1 "Introduction" about the effects on PMTUD when ICMPv6 messages are blocked.

- o Added Note to Introduction that document that this document doesn't cite RFC2119 and only uses lower case "should/must" language. Changed all upper case "should/must" to lower case.

- o Added a short summary to the Section 1 "Introduction" of Packetization Layer Path MTU Discovery ((PLPMTUD) and a reference to RFC4821 that defines it.

- o Aligned text in Section 2 "Terminology" to match current packetization layer terminology.

- o Added clarification in Section 4 "Protocol Requirements" that nodes should validate the payload of ICMP PTB message per RFC4443, and that nodes should detect decreases in PMTU as fast as possible.

- o Remove Note from Section 4 "Protocol Requirements" about a Packet Too Big message reporting a next-hop MTU that is less than the IPv6 minimum link MTU because this was removed from [I-D.ietf-6man-rfc2460bis].

- o Added clarification in Section 5.2 "Storing PMTU information" to discard an ICMPv6 Packet Too Big message if it contains a MTU less than the IPv6 minimum link MTU.

- Added clarification Section 5.2 "Storing PMTU information" that nodes with multiple interface, Path MTU information should be stored for each link.

- Removed text in Section 5.2 "Storing PMTU information" about the RH0 routing header because it was deprecated by RFC5095.

- Removed text about obsolete security classification from Section 5.2 "Storing PMTU information".

- Changed title of Section 5.4 to "Packetization Layer actions" and changed to text in the first paragraph to to generalize this section to cover all packetization layers, not just TCP.

- Clarified text in Section 5.4 "Packetization Layer actions" to use normal packetization layer retransmission methods.

- Removed text in Section 5.4 "Packetization Layer actions" that described 4.2 BSD because it is obsolete, and removed reference to TP4.

- Updated text in Section 5.5 "Issues for other transport protocols" about NFS including adding a current reference to NFS and removing obsolete text.

- Added paragraph to Section 6 "Security Considerations" about black hole connections if PTB messages are not received, and comparison to PLPMTD.

- Updated Section 7 "Acknowledgements".

- Editorial Changes.

B.1. Change History Since RFC1981

NOTE TO RFC EDITOR: Please remove this subsection prior to RFC Publication

This section describes change history made in each Internet Draft that went into producing this version. The numbers identify the Internet-Draft version in which the change was made.

Working Group Internet Drafts

08) Based on IESG comments, cleaned up text in Section 5.3 regarding suggested action when PMTU value has not been decreased recently.

08) Revision of Note in Section 5.4 to make text clearer.

08) Updated Section 7 "Acknowledgements".

08) Editorial Changes.

07) Changes from the IESG Discuss comments from IESG reviews. The changes include:

- Added Note to Introduction that document that this document doesn't cite RFC2119 and only uses lower case "should/must" language. Changed all upper case "should/must" to lower case.

- Added references for EMTU_S and EMTU_R.

- Added clarification to Section 4 "Protocol Requirements" that nodes should detect decreases in PMTU as fast as possible.

- Added clarification Section 5.2 "Storing PMTU information" that nodes with multiple interface, Path MTU information should be stored for each link.

- Removed text in Section 5.2 about Retransmission because it was unneeded.

- Removed text in Section 5.3 about Retransmission because it was unneeded.

- Rewrote text in Section 5.4 "Packetization Layer actions" regarding reception to make it clearer.

- Rewrote the text at the end of Section 5.4 to remove unnecessary details and clarify not change congestion window.

- Added references in Section 5.5 for SCTP and added DCCP (and reference) the list of examples.

- o Added paragraph to Section 5.5 "Security Considerations" about black hole connections if PTB messages are not received, and comparison to PLPMTD.

07) Editorial changes.

06) Revised Appendix B "Changes since RFC1981" to have a summary of changes since RFC1981 and a separate subsection with a change history of each Internet Draft. This subsection will be removed when the RFC is published.

06) Editorial changes based on comments received after publishing the -05 draft.

05) Changes based on IETF last call reviews by Gorry Fairhurst, Joe Touch, Susan Hares, Stewart Bryant, Rifaat Shekh-Yusef, and Donald Eastlake. This includes includes:

- o Clarify that the purpose of PMTUD is to reduce the need for IPv6 Fragmentation.
- o Added text to Introduction about effects on PMTUD when ICMPv6 messages are blocked.
- o Clarified in Section 4. that nodes should validate the payload of ICMPv6 PTB messages per RFC4443.
- o Removed text in Section 5.2 about the number of paths to a destination.
- o Changed title of Section 5.4 to "Packetization layer actions".
- o Clarified first paragraph in Section 5.4 to to cover all packetization layers, not just TCP.
- o Clarified text in Section 5.4 to use normal retransmission methods.
- o Add clarification to Note in Section 5.4 about retransmissions.
- o Removed text in Section 5.4 that described 4.2BSD as it is now obsolete.
- o Removed reference to TP4 in Section 5.5.

- o Updated text in Section 5.5 about NFS including adding a current reference to NFS and removing obsolete text.

- o Revised text in Section 6 to clarify first attack response.

- o Added new text in Section 6 to clarify the effect of ICMPv6 filtering on PMTUD.

- o Aligned terminology for the packetization layer terminology.

- o Editorial changes.

04) Changes based on AD Evaluation including removing details about RFC4821 algorithm in Section 1, remove text about decrementing hop limit from Section 3, and removed text about obsolete security classifications from Section 5.2.

04) Editorial changes and clarification in Section 5.2 based on IP Directorate review by Donald Eastlake

03) Remove text in Section 5.3 regarding RH0 since it was deprecated by RFC5095

02) Clarified in Section 3 that ICMPv6 Packet Too Big should be sent even if the node doesn’t decrement the hop limit

01) Revised the text about PLPMTUD to use the word "path".

01) Editorial changes.

00) Added text to discard an ICMPv6 Packet Too Big message containing an MTU less than the IPv6 minimum link MTU.

00) Revision of text regarding RFC4821.

00) Added R. Hinden as Editor to facilitate ID submission.

00) Editorial changes.

Individual Internet Drafts

01) Remove Note about a Packet Too Big message reporting a next-hop MTU that is less than the IPv6 minimum link MTU. This was removed from [I-D.ietf-6man-rfc2460bis].

01) Include a link to RFC4821 along with a short summary of what it does.

01) Assigned references to informative and normative.

01) Editorial changes.

00) Establish a baseline from RFC1981. The only intended changes are formatting (XML is slightly different from .nroff), differences between an RFC and Internet Draft, fixing a few ID Nits, updating references, and updates to the authors information. There should not be any content changes to the specification.

Authors' Addresses

Jack McCann
Digital Equipment Corporation

Stephen E. Deering
Retired
Vancouver, British Columbia
Canada

Jeffrey Mogul
Digital Equipment Corporation

Robert M. Hinden (editor)
Check Point Software
959 Skyway Road
San Carlos, CA 94070
USA

Email: bob.hinden@gmail.com

Network Working Group S. Deering
Internet-Draft Retired
Obsoletes: 2460 (if approved) R. Hinden
Intended status: Standards Track Check Point Software
Expires: November 20, 2017 May 19, 2017

Internet Protocol, Version 6 (IPv6) Specification
draft-ietf-6man-rfc2460bis-13

Abstract

This document specifies version 6 of the Internet Protocol (IPv6). It obsoletes RFC2460

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

IP version 6 (IPv6) is a new version of the Internet Protocol (IP), designed as the successor to IP version 4 (IPv4) [RFC0791]. The changes from IPv4 to IPv6 fall primarily into the following categories:

- Expanded Addressing Capabilities

  IPv6 increases the IP address size from 32 bits to 128 bits, to support more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler auto-configuration of addresses. The scalability of multicast routing is improved by adding a "scope" field to multicast addresses. And a new type of address called an "anycast address" is defined, used to send a packet to any one of a group of nodes.

- Header Format Simplification

  Some IPv4 header fields have been dropped or made optional, to reduce the common-case processing cost of packet handling and to limit the bandwidth cost of the IPv6 header.

- Improved Support for Extensions and Options

  Changes in the way IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future.

- Flow Labeling Capability

  A new capability is added to enable the labeling of sequences of packets that the sender requests to be treated in the network as a single flow.

- Authentication and Privacy Capabilities

  Extensions to support authentication, data integrity, and (optional) data confidentiality are specified for IPv6.

This document specifies the basic IPv6 header and the initially-defined IPv6 extension headers and options. It also discusses packet size issues, the semantics of flow labels and traffic classes, and the effects of IPv6 on upper-layer protocols. The format and semantics of IPv6 addresses are specified separately in [RFC4291].

The IPv6 version of ICMP, which all IPv6 implementations are required to include, is specified in [RFC4443]

The data transmission order for IPv6 is the same as for IPv4 as defined in Appendix B of [RFC0791].

Note: As this document obsoletes [RFC2460], any document referenced in this document that includes pointers to RFC2460, should be interpreted as referencing this document.

2. Terminology

node — a device that implements IPv6.

router — a node that forwards IPv6 packets not explicitly addressed to itself. [See Note below].

host — any node that is not a router. [See Note below].

upper layer — a protocol layer immediately above IPv6. Examples are transport protocols such as TCP and UDP, control protocols such as ICMP, routing protocols such as OSPF, and internet or lower-layer protocols being "tunneled" over (i.e., encapsulated in) IPv6 such as IPX, AppleTalk, or IPv6 itself.

link — a communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IPv6. Examples are Ethernets (simple or bridged); PPP links; X.25, Frame Relay, or ATM networks; and internet (or higher) layer "tunnels", such as tunnels over IPv4 or IPv6 itself.

neighbors — nodes attached to the same link.

interface — a node’s attachment to a link.

address — an IPv6-layer identifier for an interface or a set of interfaces.

packet — an IPv6 header plus payload.

link MTU — the maximum transmission unit, i.e., maximum packet size in octets, that can be conveyed over a link.

path MTU — the minimum link MTU of all the links in a path between a source node and a destination node.

Note: it is possible for a device with multiple interfaces to be configured to forward non-self-destined packets arriving from some set (fewer than all) of its interfaces, and to discard non-self-destined packets arriving from its other interfaces. Such a device must obey the protocol requirements for routers when receiving packets from, and interacting with neighbors over, the former (forwarding) interfaces. It must obey the protocol requirements for hosts when receiving packets from, and interacting with neighbors over, the latter (non-forwarding) interfaces.

3. IPv6 Header Format

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version — 4-bit Internet Protocol version number = 6.

Traffic Class — 8-bit traffic class field. See section 7.

Flow Label — 20-bit flow label. See section 6.

Payload Length — 16-bit unsigned integer. Length of the IPv6 payload, i.e., the rest of the packet following this IPv6 header, in octets. (Note that any extension headers [Section 4] present are considered part of the payload, i.e., included in the length count.)

Next Header — 8-bit selector. Identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 Protocol field [IANA-PN].

Hop Limit — 8-bit unsigned integer. Decremented by 1 by each node that forwards the packet. When forwarding, the packet is discarded if Hop Limit was zero when received or is decremented to zero. A node that is the destination of a packet should not discard a packet with hop limit equal to zero, it should process the packet normally.

Source Address — 128-bit address of the originator of the packet. See [RFC4291].

Destination Address — 128-bit address of the intended recipient of the packet (possibly not the ultimate recipient, if a Routing header is present). See [RFC4291] and section 4.4.

4. IPv6 Extension Headers

In IPv6, optional internet-layer information is encoded in separate headers that may be placed between the IPv6 header and the upper-layer header in a packet. There is a small number of such extension headers, each one identified by a distinct Next Header value.

Extension Headers are numbered from IANA IP Protocol Numbers [IANA-PN], the same values used for IPv4 and IPv6. When processing a sequence of Next Header values in a packet, the first one that is not an Extension Header [IANA-EH] indicates that the next item in the packet is the corresponding upper-layer header. A special "No Next Header" value is used if there is no upper-layer header.

As illustrated in these examples, an IPv6 packet may carry zero, one, or more extension headers, each identified by the Next Header field of the preceding header:

```
+---------------+------------------------
|  IPv6 header  | TCP header + data
|               |
| Next Header = |
|      TCP      |
+---------------+------------------------

+---------------+----------------+------------------------
|  IPv6 header  | Routing header | TCP header + data
|               |                |
| Next Header = |  Next Header = |
|    Routing    |      TCP       |
+---------------+----------------+------------------------

+---------------+----------------+-----------------+-----------------
|  IPv6 header  | Routing header | Fragment header | fragment of TCP
|               |                |                 |  header + data
| Next Header = |  Next Header = |  Next Header =  |
|    Routing    |    Fragment    |       TCP       |
+---------------+----------------+-----------------+-----------------
```

Extension headers (except for the Hop-by-Hop Options header) are not processed, inserted, or deleted by any node along a packet's delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header.

The Hop-by-Hop Options header is not inserted or deleted, but may be examined or processed by any node along a packet's delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header. The Hop-by-Hop Options header, when present, must immediately follow the IPv6 header. Its presence is indicated by the value zero in the Next Header field of the IPv6 header.

NOTE: While [RFC2460] required that all nodes must examine and process the Hop-by-Hop Options header, it is now expected that nodes along a packet's delivery path only examine and process the Hop-by-Hop Options header if explicitly configured to do so.

At the Destination node, normal demultiplexing on the Next Header field of the IPv6 header invokes the module to process the first extension header, or the upper-layer header if no extension header is present. The contents and semantics of each extension header determine whether or not to proceed to the next header. Therefore, extension headers must be processed strictly in the order they appear in the packet; a receiver must not, for example, scan through a packet looking for a particular kind of extension header and process that header prior to processing all preceding ones.

If, as a result of processing a header, the destination node is required to proceed to the next header but the Next Header value in the current header is unrecognized by the node, it should discard the packet and send an ICMP Parameter Problem message to the source of the packet, with an ICMP Code value of 1 ("unrecognized Next Header type encountered") and the ICMP Pointer field containing the offset of the unrecognized value within the original packet. The same action should be taken if a node encounters a Next Header value of zero in any header other than an IPv6 header.

Each extension header is an integer multiple of 8 octets long, in order to retain 8-octet alignment for subsequent headers. Multi-octet fields within each extension header are aligned on their natural boundaries, i.e., fields of width n octets are placed at an integer multiple of n octets from the start of the header, for n = 1, 2, 4, or 8.

A full implementation of IPv6 includes implementation of the following extension headers:

Hop-by-Hop Options
Fragment
Destination Options
Routing
Authentication
Encapsulating Security Payload

The first four are specified in this document; the last two are specified in [RFC4302] and [RFC4303], respectively. The current list of IPv6 extension headers can be found at [IANA-EH].

4.1. Extension Header Order

When more than one extension header is used in the same packet, it is recommended that those headers appear in the following order:

IPv6 header
Hop-by-Hop Options header
Destination Options header (note 1)
Routing header
Fragment header
Authentication header (note 2)
Encapsulating Security Payload header (note 2)
Destination Options header (note 3)
upper-layer header

note 1: for options to be processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header.

note 2: additional recommendations regarding the relative order of the Authentication and Encapsulating Security Payload headers are given in [RFC4303].

note 3: for options to be processed only by the final destination of the packet.

Each extension header should occur at most once, except for the Destination Options header which should occur at most twice (once before a Routing header and once before the upper-layer header).

If the upper-layer header is another IPv6 header (in the case of IPv6 being tunneled over or encapsulated in IPv6), it may be followed by its own extension headers, which are separately subject to the same ordering recommendations.

If and when other extension headers are defined, their ordering constraints relative to the above listed headers must be specified.

IPv6 nodes must accept and attempt to process extension headers in any order and occurring any number of times in the same packet, except for the Hop-by-Hop Options header which is restricted to appear immediately after an IPv6 header only. Nonetheless, it is strongly advised that sources of IPv6 packets adhere to the above recommended order until and unless subsequent specifications revise that recommendation.

4.2. Options

Two of the currently-defined extension headers defined in this document -- the Hop-by-Hop Options header and the Destination Options header -- carry a variable number of type-length-value (TLV) encoded "options", of the following format:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
|  Option Type  |  Opt Data Len |  Option Data
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
```

Option Type — 8-bit identifier of the type of option.

Opt Data Len — 8-bit unsigned integer. Length of the Option Data field of this option, in octets.

Option Data Variable-length field. Option-Type-specific data.

The sequence of options within a header must be processed strictly in the order they appear in the header; a receiver must not, for example, scan through the header looking for a particular kind of option and process that option prior to processing all preceding ones.

The Option Type identifiers are internally encoded such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type:

00 - skip over this option and continue processing the header.

01 - discard the packet.

10 - discard the packet and, regardless of whether or not the packet’s Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet’s Source Address, pointing to the unrecognized Option Type.

11 - discard the packet and, only if the packet’s Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet’s Source Address, pointing to the unrecognized Option Type.

The third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en-route to the packet’s final destination. When an Authentication header is present in the packet, for any option whose data may change en-route, its entire Option Data field must be treated as zero-valued octets when computing or verifying the packet’s authenticating value.

0 - Option Data does not change en-route

1 - Option Data may change en-route

The three high-order bits described above are to be treated as part of the Option Type, not independent of the Option Type. That is, a particular option is identified by a full 8-bit Option Type, not just the low-order 5 bits of an Option Type.

The same Option Type numbering space is used for both the Hop-by-Hop Options header and the Destination Options header. However, the specification of a particular option may restrict its use to only one of those two headers.

Individual options may have specific alignment requirements, to ensure that multi-octet values within Option Data fields fall on natural boundaries. The alignment requirement of an option is specified using the notation xn+y, meaning the Option Type must appear at an integer multiple of x octets from the start of the header, plus y octets. For example:

```
2n   means any 2-octet offset from the start of the header.
8n+2 means any 8-octet offset from the start of the header, plus 2
     octets.
```

There are two padding options which are used when necessary to align subsequent options and to pad out the containing header to a multiple of 8 octets in length. These padding options must be recognized by all IPv6 implementations:

Pad1 option (alignment requirement: none)

```
+-+-+-+-+-+-+-+-+
|       0       |
+-+-+-+-+-+-+-+-+
```

NOTE! the format of the Pad1 option is a special case -- it does not have length and value fields.

The Pad1 option is used to insert one octet of padding into the Options area of a header. If more than one octet of padding is required, the PadN option, described next, should be used, rather than multiple Pad1 options.

PadN option (alignment requirement: none)

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
|       1       |  Opt Data Len |  Option Data
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
```

The PadN option is used to insert two or more octets of padding into the Options area of a header. For N octets of padding, the

Opt Data Len field contains the value N-2, and the Option Data consists of N-2 zero-valued octets.

Appendix A contains formatting guidelines for designing new options.

4.3. Hop-by-Hop Options Header

The Hop-by-Hop Options header is used to carry optional information that may be examined and processed by every node along a packet's delivery path. The Hop-by-Hop Options header is identified by a Next Header value of 0 in the IPv6 header, and has the following format:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |  Hdr Ext Len  |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
.                                                               .
.                            Options                            .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Next Header — 8-bit selector. Identifies the type of header immediately following the Hop-by-Hop Options header. Uses the same values as the IPv4 Protocol field [IANA-PN].

Hdr Ext Len — 8-bit unsigned integer. Length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets.

Options — Variable-length field, of length such that the complete Hop-by-Hop Options header is an integer multiple of 8 octets long. Contains one or more TLV-encoded options, as described in section 4.2.

The only hop-by-hop options defined in this document are the Pad1 and PadN options specified in section 4.2.

4.4. Routing Header

The Routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination. This function is very similar to IPv4's Loose Source and Record Route option. The Routing header is identified by a Next Header value of 43 in the immediately preceding header, and has the following format:

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Next Header  |  Hdr Ext Len  |  Routing Type | Segments Left |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 .                                                               .
 .                       type-specific data                      .
 .                                                               .
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Next Header	8-bit selector. Identifies the type of header immediately following the Routing header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Routing header in 8-octet units, not including the first 8 octets.
Routing Type	8-bit identifier of a particular Routing header variant.
Segments Left	8-bit unsigned integer. Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination.
type-specific data	Variable-length field, of format determined by the Routing Type, and of length such that the complete Routing header is an integer multiple of 8 octets long.

If, while processing a received packet, a node encounters a Routing header with an unrecognized Routing Type value, the required behavior of the node depends on the value of the Segments Left field, as follows:

If Segments Left is zero, the node must ignore the Routing header and proceed to process the next header in the packet, whose type is identified by the Next Header field in the Routing header.

If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

If, after processing a Routing header of a received packet, an intermediate node determines that the packet is to be forwarded onto a link whose link MTU is less than the size of the packet, the node must discard the packet and send an ICMP Packet Too Big message to the packet's Source Address.

The currently defined IPv6 Routing Headers and their status can be found at [IANA-RH]. Allocation guidelines for IPv6 Routing Headers can be found in [RFC5871].

4.5. Fragment Header

The Fragment header is used by an IPv6 source to send a packet larger than would fit in the path MTU to its destination. (Note: unlike IPv4, fragmentation in IPv6 is performed only by source nodes, not by routers along a packet's delivery path -- see section 5.) The Fragment header is identified by a Next Header value of 44 in the immediately preceding header, and has the following format:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |   Reserved    |      Fragment Offset    |Res|M|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Identification                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Next Header	8-bit selector. Identifies the initial header type of the Fragmentable Part of the original packet (defined below). Uses the same values as the IPv4 Protocol field [IANA-PN].
Reserved	8-bit reserved field. Initialized to zero for transmission; ignored on reception.
Fragment Offset	13-bit unsigned integer. The offset, in 8-octet units, of the data following this header, relative to the start of the Fragmentable Part of the original packet.
Res	2-bit reserved field. Initialized to zero for transmission; ignored on reception.
M flag	1 = more fragments; 0 = last fragment.

Identification 32 bits. See description below.

In order to send a packet that is too large to fit in the MTU of the path to its destination, a source node may divide the packet into fragments and send each fragment as a separate packet, to be reassembled at the receiver.

For every packet that is to be fragmented, the source node generates an Identification value. The Identification must be different than that of any other fragmented packet sent recently* with the same Source Address and Destination Address. If a Routing header is present, the Destination Address of concern is that of the final destination.

* "recently" means within the maximum likely lifetime of a packet, including transit time from source to destination and time spent awaiting reassembly with other fragments of the same packet. However, it is not required that a source node knows the maximum packet lifetime. Rather, it is assumed that the requirement can be met by implementing an algorithm that results in a low identification reuse frequency. Examples of algorithms that can meet this requirement are described in [RFC7739].

The initial, large, unfragmented packet is referred to as the "original packet", and it is considered to consist of three parts, as illustrated:

original packet:

```
+------------------+-------------------------+---//----------------+
|  Per-Fragment    | Extension & Upper-Layer |   Fragmentable       |
|    Headers       |        Headers          |       Part          |
+------------------+-------------------------+---//----------------+
```

The Per-Fragment Headers must consist of the IPv6 header plus any extension headers that must be processed by nodes en route to the destination, that is, all headers up to and including the Routing header if present, else the Hop-by-Hop Options header if present, else no extension headers.

The Extension Headers are all other extension headers that are not included in the Per-Fragment headers part of the packet. For this purpose, the Encapsulating Security Payload (ESP) is not considered an extension header. The Upper-Layer Header is the first upper-layer header that is not an IPv6 extension header.

Examples of upper-layer headers include TCP, UDP, IPv4, IPv6, ICMPv6, and as noted ESP.

The Fragmentable Part consists of the rest of the packet after the upper-layer header or after any header (i.e., initial IPv6 header or extension header) that contains a Next Header value of No Next Header.

The Fragmentable Part of the original packet is divided into fragments. The lengths of the fragments must be chosen such that the resulting fragment packets fit within the MTU of the path to the packets' destination(s). Each complete fragment, except possibly the last ("rightmost") one, being an integer multiple of 8 octets long.

The fragments are transmitted in separate "fragment packets" as illustrated:

original packet:

```
+-----------------+-----------------+--------+--------+-//-+--------+
|  Per-Fragment   |Ext & Upper-Layer|  first | second |    |  last  |
|    Headers      |    Headers      |fragment|fragment|....|fragment|
+-----------------+-----------------+--------+--------+-//-+--------+
```

fragment packets:

```
+------------------+---------+-------------------+----------+
|  Per-Fragment    |Fragment | Ext & Upper-Layer |  first   |
|    Headers       | Header  |   Headers         | fragment |
+------------------+---------+-------------------+----------+

+------------------+--------+-------------------------------+
|  Per-Fragment    |Fragment|    second                     |
|    Headers       | Header |   fragment                    |
+------------------+--------+-------------------------------+
                      o
                      o
                      o
+------------------+--------+----------+
|  Per-Fragment    |Fragment|   last   |
|    Headers       | Header | fragment |
+------------------+--------+----------+
```

The first fragment packet is composed of:

(1) The Per-Fragment Headers of the original packet, with the Payload Length of the original IPv6 header changed to contain the length of this fragment packet only (excluding the length of the

IPv6 header itself), and the Next Header field of the last header of the Per-Fragment Headers changed to 44.

(2) A Fragment header containing:

The Next Header value that identifies the first header after the Per-Fragment Headers of the original packet.

A Fragment Offset containing the offset of the fragment, in 8-octet units, relative to the start of the Fragmentable Part of the original packet. The Fragment Offset of the first ("leftmost") fragment is 0.

An M flag value of 1 as this is the first fragment.

The Identification value generated for the original packet.

(3) Extension Headers, if any, and the Upper-Layer header. These headers must be in the first fragment. Note: This restricts the size of the headers through the Upper-Layer header to the MTU of the path to the packets’ destinations(s).

(4) The first fragment.

The subsequent fragment packets are composed of:

(1) The Per-Fragment Headers of the original packet, with the Payload Length of the original IPv6 header changed to contain the length of this fragment packet only (excluding the length of the IPv6 header itself), and the Next Header field of the last header of the Per-Fragment Headers changed to 44.

(2) A Fragment header containing:

The Next Header value that identifies the first header after the Per-Fragment Headers of the original packet.

A Fragment Offset containing the offset of the fragment, in 8-octet units, relative to the start of the Fragmentable part of the original packet.

An M flag value of 0 if the fragment is the last ("rightmost") one, else an M flag value of 1.

The Identification value generated for the original packet.

(3) The fragment itself.

Fragments must not be created that overlap with any other fragments created from the original packet.

At the destination, fragment packets are reassembled into their original, unfragmented form, as illustrated:

reassembled original packet:

```
+---------------+----------------+---------+--------+-//--+--------+
| Per-Fragment  |Ext & Upper-Layer|  first  | second |      | last   |
|    Headers    |   Headers      |frag data|fragment|.....|fragment|
+---------------+----------------+---------+--------+-//--+--------+
```

The following rules govern reassembly:

An original packet is reassembled only from fragment packets that have the same Source Address, Destination Address, and Fragment Identification.

The Per-Fragment Headers of the reassembled packet consists of all headers up to, but not including, the Fragment header of the first fragment packet (that is, the packet whose Fragment Offset is zero), with the following two changes:

The Next Header field of the last header of the Per-Fragment Headers is obtained from the Next Header field of the first fragment's Fragment header.

The Payload Length of the reassembled packet is computed from the length of the Per-Fragment Headers and the length and offset of the last fragment. For example, a formula for computing the Payload Length of the reassembled original packet is:

```
PL.orig = PL.first - FL.first - 8 + (8 * FO.last) + FL.last
```

```
where
PL.orig  =  Payload Length field of reassembled packet.
PL.first =  Payload Length field of first fragment packet.
```

```
FL.first =  length of fragment following Fragment header of
            first fragment packet.
FO.last  =  Fragment Offset field of Fragment header of last
            fragment packet.
FL.last  =  length of fragment following Fragment header of
            last fragment packet.
```

The Fragmentable Part of the reassembled packet is constructed from the fragments following the Fragment headers in each of the fragment packets. The length of each fragment is computed by subtracting from the packet's Payload Length the length of the headers between the IPv6 header and fragment itself; its relative position in Fragmentable Part is computed from its Fragment Offset value.

The Fragment header is not present in the final, reassembled packet.

If the fragment is a whole datagram (that is, both the Fragment Offset field and the M flag are zero), then it does not need any further reassembly and should be processed as a fully reassembled packet (i.e., updating Next Header, adjust Payload Length, removing the Fragmentation Header, etc.). Any other fragments that match this packet (i.e., the same IPv6 Source Address, IPv6 Destination Address, and Fragment Identification) should be processed independently.

The following error conditions may arise when reassembling fragmented packets:

- If insufficient fragments are received to complete reassembly of a packet within 60 seconds of the reception of the first-arriving fragment of that packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded. If the first fragment (i.e., the one with a Fragment Offset of zero) has been received, an ICMP Time Exceeded -- Fragment Reassembly Time Exceeded message should be sent to the source of that fragment.

- If the length of a fragment, as derived from the fragment packet's Payload Length field, is not a multiple of 8 octets and the M flag of that fragment is 1, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Payload Length field of the fragment packet.

- o If the length and offset of a fragment are such that the Payload Length of the packet reassembled from that fragment would exceed 65,535 octets, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Fragment Offset field of the fragment packet.

- o If the first fragment does not include all headers through an Upper-Layer header, then that fragment should be discarded and an ICMP Parameter Problem, Code 3, message should be sent to the source of the fragment, with the Pointer field set to zero.

- o If any of the fragments being reassembled overlaps with any other fragments being reassembled for the same packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded and no ICMP error messages should be sent.

  It should be noted that fragments may be duplicated in the network. Instead of treating these exact duplicate fragments as overlapping fragments, an implementation may choose to detect this case and drop exact duplicate fragments while keeping the other fragments belonging to the same packet.

The following conditions are not expected to occur frequently, but are not considered errors if they do:

The number and content of the headers preceding the Fragment header of different fragments of the same original packet may differ. Whatever headers are present, preceding the Fragment header in each fragment packet, are processed when the packets arrive, prior to queueing the fragments for reassembly. Only those headers in the Offset zero fragment packet are retained in the reassembled packet.

The Next Header values in the Fragment headers of different fragments of the same original packet may differ. Only the value from the Offset zero fragment packet is used for reassembly.

Other fields in the IPv6 header may also vary across the fragments being reassembled. Specifications that use these fields may provide additional instructions if the basic mechanism of using the values from the Offset zero fragment is not sufficient. For example, Section 5.3 of [RFC3168] describes how to combine the Explicit Congestion Notification (ECN) bits from different fragments to derive the ECN bits of the reassembled packet.

4.6. Destination Options Header

The Destination Options header is used to carry optional information that need be examined only by a packet's destination node(s). The Destination Options header is identified by a Next Header value of 60 in the immediately preceding header, and has the following format:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |  Hdr Ext Len  |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
.                                                               .
.                            Options                            .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Next Header — 8-bit selector. Identifies the type of header immediately following the Destination Options header. Uses the same values as the IPv4 Protocol field [IANA-PN].

Hdr Ext Len — 8-bit unsigned integer. Length of the Destination Options header in 8-octet units, not including the first 8 octets.

Options — Variable-length field, of length such that the complete Destination Options header is an integer multiple of 8 octets long. Contains one or more TLV-encoded options, as described in section 4.2.

The only destination options defined in this document are the Pad1 and PadN options specified in section 4.2.

Note that there are two possible ways to encode optional destination information in an IPv6 packet: either as an option in the Destination Options header, or as a separate extension header. The Fragment header and the Authentication header are examples of the latter approach. Which approach can be used depends on what action is desired of a destination node that does not understand the optional information:

- o If the desired action is for the destination node to discard the packet and, only if the packet's Destination Address is not a multicast address, send an ICMP Unrecognized Type message to the packet's Source Address, then the information may be encoded either as a separate header or as an option in the Destination Options header whose Option Type has the value 11 in its highest-order two bits. The choice may depend on such factors as which takes fewer octets, or which yields better alignment or more efficient parsing.

- o If any other action is desired, the information must be encoded as an option in the Destination Options header whose Option Type has the value 00, 01, or 10 in its highest-order two bits, specifying the desired action (see section 4.2).

4.7. No Next Header

The value 59 in the Next Header field of an IPv6 header or any extension header indicates that there is nothing following that header. If the Payload Length field of the IPv6 header indicates the presence of octets past the end of a header whose Next Header field contains 59, those octets must be ignored, and passed on unchanged if the packet is forwarded.

4.8. Defining New Extension Headers and Options

Defining new IPv6 extension headers is not recommended, unless there are no existing IPv6 extension headers that can be used by specifying a new option for that IPv6 extension header. A proposal to specify a new IPv6 extension header must include a detailed technical explanation of why an existing IPv6 extension header can not be used for the desired new function. See [RFC6564] for additional background information.

Note: New extension headers that require hop-by-hop behavior must not be defined because, as specified in Section 4 of this document, the only Extension Header that has hop-by-hop behavior is the Hop-by-Hop Options header.

New hop-by-hop options are not recommended because nodes may be configured to ignore the Hop-by-Hop Option header, drop packets containing a hop-by-hop header, or assign packets containing a hop-by-hop header to a slow processing path. Designers considering defining new hop-by-hop options need to be aware of this likely behaviour. There has to be a very clear justification why any new hop-by-hop option is needed before it is standardized.

Instead of defining new Extension Headers, it is recommended that the Destination Options header is used to carry optional information that must be examined only by a packet's destination node(s), because they provide better handling and backward compatibility.

If new Extension Headers are defined, they need to use the following format:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |  Hdr Ext Len  |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
.                                                               .
.                  Header Specific Data                         .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Next Header 8-bit selector. Identifies the type of header immediately following the extension header. Uses the same values as the IPv4 Protocol field [IANA-PN].

Hdr Ext Len 8-bit unsigned integer. Length of the Destination Options header in 8-octet units, not including the first 8 octets.

Header Specific Data Variable-length field. Fields specific to the extension header.

5. Packet Size Issues

IPv6 requires that every link in the internet have an MTU of 1280 octets or greater. This is known as the IPv6 minimum link MTU. On any link that cannot convey a 1280-octet packet in one piece, link-specific fragmentation and reassembly must be provided at a layer below IPv6.

Links that have a configurable MTU (for example, PPP links [RFC1661]) must be configured to have an MTU of at least 1280 octets; it is recommended that they be configured with an MTU of 1500 octets or greater, to accommodate possible encapsulations (i.e., tunneling) without incurring IPv6-layer fragmentation.

From each link to which a node is directly attached, the node must be able to accept packets as large as that link's MTU.

It is strongly recommended that IPv6 nodes implement Path MTU Discovery [RFC1981], in order to discover and take advantage of path MTUs greater than 1280 octets. However, a minimal IPv6 implementation (e.g., in a boot ROM) may simply restrict itself to sending packets no larger than 1280 octets, and omit implementation of Path MTU Discovery.

In order to send a packet larger than a path's MTU, a node may use the IPv6 Fragment header to fragment the packet at the source and have it reassembled at the destination(s). However, the use of such fragmentation is discouraged in any application that is able to adjust its packets to fit the measured path MTU (i.e., down to 1280 octets).

A node must be able to accept a fragmented packet that, after reassembly, is as large as 1500 octets. A node is permitted to accept fragmented packets that reassemble to more than 1500 octets. An upper-layer protocol or application that depends on IPv6 fragmentation to send packets larger than the MTU of a path should not send packets larger than 1500 octets unless it has assurance that the destination is capable of reassembling packets of that larger size.

6. Flow Labels

The 20-bit Flow Label field in the IPv6 header is used by a source to label sequences of packets to be treated in the network as a single flow.

The current definition of the IPv6 Flow Label can be found in [RFC6437].

7. Traffic Classes

The 8-bit Traffic Class field in the IPv6 header is used by the network for traffic management. The value of the Traffic Class bits in a received packet or fragment might be different from the value sent by the packet's source.

The current use of the Traffic Class field for Differentiated Services and Explicit Congestion Notification is specified in [RFC2474] and [RFC3168].

8. Upper-Layer Protocol Issues

8.1. Upper-Layer Checksums

Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6, to include the 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. In particular, the following illustration shows the TCP and UDP "pseudo-header" for IPv6:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Upper-Layer Packet Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      zero                     |  Next Header  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- o If the IPv6 packet contains a Routing header, the Destination Address used in the pseudo-header is that of the final destination. At the originating node, that address will be in the last element of the Routing header; at the recipient(s), that address will be in the Destination Address field of the IPv6 header.

- o The Next Header value in the pseudo-header identifies the upper-layer protocol (e.g., 6 for TCP, or 17 for UDP). It will differ from the Next Header value in the IPv6 header if there are extension headers between the IPv6 header and the upper-layer header.

- o The Upper-Layer Packet Length in the pseudo-header is the length of the upper-layer header and data (e.g., TCP header plus TCP data). Some upper-layer protocols carry their own length information (e.g., the Length field in the UDP header); for such protocols, that is the length used in the pseudo-header. Other protocols (such as TCP) do not carry their own length information, in which case the length used in the pseudo-header is the Payload Length from the IPv6 header, minus the length of any extension headers present between the IPv6 header and the upper-layer header.

- o Unlike IPv4, the default behavior when UDP packets are originated by an IPv6 node is that the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error.

- o As an exception to the default behaviour, protocols that use UDP as a tunnel encapsulation may enable zero-checksum mode for a specific port (or set of ports) for sending and/or receiving. Any node implementing zero-checksum mode must follow the requirements specified in "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums" [RFC6936].

The IPv6 version of ICMP [RFC4443] includes the above pseudo-header in its checksum computation; this is a change from the IPv4 version of ICMP, which does not include a pseudo-header in its checksum. The reason for the change is to protect ICMP from misdelivery or corruption of those fields of the IPv6 header on which it depends, which, unlike IPv4, are not covered by an internet-layer checksum. The Next Header field in the pseudo-header for ICMP contains the value 58, which identifies the IPv6 version of ICMP.

8.2. Maximum Packet Lifetime

Unlike IPv4, IPv6 nodes are not required to enforce maximum packet lifetime. That is the reason the IPv4 "Time to Live" field was renamed "Hop Limit" in IPv6. In practice, very few, if any, IPv4 implementations conform to the requirement that they limit packet lifetime, so this is not a change in practice. Any upper-layer protocol that relies on the internet layer (whether IPv4 or IPv6) to limit packet lifetime ought to be upgraded to provide its own mechanisms for detecting and discarding obsolete packets.

8.3. Maximum Upper-Layer Payload Size

When computing the maximum payload size available for upper-layer data, an upper-layer protocol must take into account the larger size of the IPv6 header relative to the IPv4 header. For example, in IPv4, TCP’s MSS option is computed as the maximum packet size (a default value or a value learned through Path MTU Discovery) minus 40 octets (20 octets for the minimum-length IPv4 header and 20 octets for the minimum-length TCP header). When using TCP over IPv6, the MSS must be computed as the maximum packet size minus 60 octets, because the minimum-length IPv6 header (i.e., an IPv6 header with no extension headers) is 20 octets longer than a minimum-length IPv4 header.

8.4. Responding to Packets Carrying Routing Headers

When an upper-layer protocol sends one or more packets in response to a received packet that included a Routing header, the response packet(s) must not include a Routing header that was automatically derived by "reversing" the received Routing header UNLESS the integrity and authenticity of the received Source Address and Routing header have been verified (e.g., via the use of an Authentication header in the received packet). In other words, only the following kinds of packets are permitted in response to a received packet bearing a Routing header:

- o Response packets that do not carry Routing headers.

- o Response packets that carry Routing headers that were NOT derived by reversing the Routing header of the received packet (for example, a Routing header supplied by local configuration).

- o Response packets that carry Routing headers that were derived by reversing the Routing header of the received packet IF AND ONLY IF the integrity and authenticity of the Source Address and Routing header from the received packet have been verified by the responder.

9. IANA Considerations

RFC2460 is referenced in a number of IANA registries. These include:

- o Internet Protocol Version 6 (IPv6) Parameters [IANA-6P]

- o Assigned Internet Protocol Numbers [IANA-PN]

- o ONC RPC Network Identifiers (netids) [IANA-NI]

- o Technical requirements for authoritative name servers [IANA-NS]

- o Network Layer Protocol Identifiers (NLPIDs) of Interest [IANA-NL]

- o Protocol Registries [IANA-PR]

- o Structure of Management Information (SMI) Numbers (MIB Module Registrations) [IANA-MI]

The IANA should update these references to point to this document.

10. Security Considerations

IPv6, from the viewpoint of the basic format and transmission of packets, has security properties that are similar to IPv4. These security issues include:

- o Eavesdropping, On-path elements can observe the whole packet (including both contents and metadata) of each IPv6 datagram.
- o Replay, where attacker records a sequence of packets off of the wire and plays them back to the party which originally received them.
- o Packet insertion, where the attacker forges a packet with some chosen set of properties and injects it into the network.
- o Packet deletion, where the attacker remove a packet from the wire.
- o Packet modification, where the attacker removes a packet from the wire, modifies it, and re-injects it into the network.
- o Man in the Middle attacks, where the attacker subverts the communication stream in order to pose as the sender to receiver and the receiver to the sender.
- o Denial of Service Attacks, where the attacker sends large amounts of legitimate traffic to a destination to overwhelm it.

IPv6 packets can be protected from eavesdropping, replay, packet insertion, packet modification, and man in the middle attacks by use of the "Security Architecture for the Internet Protocol" [RFC4301]. In addition, upper-layer protocols such as TLS or SSH can be used to protect the application layer traffic running on top of IPv6.

There is not any mechanism to protect against "denial of service attacks". Defending against these type of attacks is outside the scope of this specification.

IPv6 addresses are significantly larger than IPv4 address making it much harder to scan the address space across the Internet and even on a single network link (e.g., Local Area Network). See [RFC7707] for more information.

IPv6 addresses of nodes are expected to be more visible on the Internet as compared with IPv4 since the use of address translation technology is reduced. This creates some additional privacy issues such as making it easier to distinguish endpoints. See [RFC7721] for more information.

The design of IPv6 extension headers architecture, while adding a lot of flexibility, also creates new security challenges. As noted below, issues relating the fragment extension header have been resolved, but it’s clear that for any new extension header designed in the future, the security implications need to be examined throughly, and this needs to include how the new extension header works with existing extension headers. See [RFC7045] for more information.

This version of the IPv6 specification resolves a number of security issues that were found with the previous version [RFC2460] of the IPv6 specification. These include:

o Revised the text to handle the case of fragments that are whole datagrams (i.e., both the Fragment Offset field and the M flag are zero). If received they should be processed as a reassembled packet. Any other fragments that match should be processed independently. The Fragment creation process was modified to not create whole datagram fragments (Fragment Offset field and the M flag are zero). See [RFC6946] and [RFC8021] for more information.

o Changed the text to require that IPv6 nodes must not create overlapping fragments. Also, when reassembling an IPv6 datagram, if one or more its constituent fragments is determined to be an overlapping fragment, the entire datagram (and any constituent fragments) must be silently discarded. Includes clarification that no ICMP error message should be sent if overlapping fragments are received. See [RFC5722] for more information.

0 Revised the text to require that all headers through the first Upper-Layer Header are in the first fragment. See [RFC6946] for more information.

o Removed the paragraph in Section 5 that required including a fragment header to outgoing packets if a ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280. See [RFC7112] for more information.

o Incorporated the updates from [RFC5095] and [RFC5871] to remove the description of the RH0 Routing Header, that the allocations guidelines for routing headers are specified in RFC5871, and removed RH0 Routing Header from the list of required extension headers.

Security issues relating to other parts of IPv6 including addressing, ICMPv6, Path MTU Discovery, etc., are discussed in the appropriate specifications.

11. Acknowledgments

The authors gratefully acknowledge the many helpful suggestions of the members of the IPng working group, the End-to-End Protocols research group, and the Internet Community At Large.

The authors would also like to acknowledge the authors of the updating RFCs that were incorporated in this version of the document to move the IPv6 specification to Internet Standard. They are Joe Abley, Shane Amante, Jari Arkko, Manav Bhatia, Ronald P. Bonica, Scott Bradner, Brian Carpenter, P.F. Chimento, Marshall Eubanks, Fernando Gont, James Hoagland, Sheng Jiang, Erik Kline, Suresh Krishnan, Vishwas Manral, George Neville-Neil, Jarno Rajahalme, Pekka Savola, Magnus Westerlund, and James Woodyatt.

12. References

12.1. Normative References

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

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

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

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

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

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

12.2. Informative References

[IANA-6P] "Internet Protocol Version 6 (IPv6) Parameters", <https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml>.

[IANA-EH] "IPv6 Extension Header Types", <https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#extension-header>.

[IANA-MI] "Structure of Management Information (SMI) Numbers (MIB Module Registrations)", < http://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml>.

[IANA-NI] "ONC RPC Network Identifiers (netids)", <http://www.iana.org/assignments/rpc-netids/rpc-netids.xhtml>.

[IANA-NL] "Network Layer Protocol Identifiers (NLPIDs) of Interest", <http://www.iana.org/assignments/nlpids/nlpids.xhtml>.

[IANA-NS] "Technical requirements for authoritative name servers", <https://www.iana.org/help/nameserver-requirements>.

[IANA-PN] "Assigned Internet Protocol Numbers", <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>.

[IANA-PR] "Protocol Registries", <https://www.iana.org/protocols>.

[IANA-RH] "IANA Routing Types Parameter Registry", <https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-3>.

[RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, DOI 10.17487/RFC1661, July 1994, <http://www.rfc-editor.org/info/rfc1661>.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <http://www.rfc-editor.org/info/rfc1981>.

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

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <http://www.rfc-editor.org/info/rfc4301>.

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

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

[RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <http://www.rfc-editor.org/info/rfc5095>.

[RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, DOI 10.17487/RFC5722, December 2009, <http://www.rfc-editor.org/info/rfc5722>.

[RFC5871] Arkko, J. and S. Bradner, "IANA Allocation Guidelines for the IPv6 Routing Header", RFC 5871, DOI 10.17487/RFC5871, May 2010, <http://www.rfc-editor.org/info/rfc5871>.

[RFC6564] Krishnan, S., Woodyatt, J., Kline, E., Hoagland, J., and M. Bhatia, "A Uniform Format for IPv6 Extension Headers", RFC 6564, DOI 10.17487/RFC6564, April 2012, <http://www.rfc-editor.org/info/rfc6564>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <http://www.rfc-editor.org/info/rfc6936>.

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

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

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

[RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", RFC 7707, DOI 10.17487/RFC7707, March 2016, <http://www.rfc-editor.org/info/rfc7707>.

[RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <http://www.rfc-editor.org/info/rfc7721>.

[RFC7739] Gont, F., "Security Implications of Predictable Fragment Identification Values", RFC 7739, DOI 10.17487/RFC7739, February 2016, <http://www.rfc-editor.org/info/rfc7739>.

[RFC8021] Gont, F., Liu, W., and T. Anderson, "Generation of IPv6 Atomic Fragments Considered Harmful", RFC 8021, DOI 10.17487/RFC8021, January 2017, <http://www.rfc-editor.org/info/rfc8021>.

Appendix A. Formatting Guidelines for Options

This appendix gives some advice on how to lay out the fields when designing new options to be used in the Hop-by-Hop Options header or the Destination Options header, as described in section 4.2. These guidelines are based on the following assumptions:

- One desirable feature is that any multi-octet fields within the Option Data area of an option be aligned on their natural boundaries, i.e., fields of width n octets should be placed at an integer multiple of n octets from the start of the Hop-by-Hop or Destination Options header, for n = 1, 2, 4, or 8.

- o Another desirable feature is that the Hop-by-Hop or Destination Options header take up as little space as possible, subject to the requirement that the header be an integer multiple of 8 octets long.

- o It may be assumed that, when either of the option-bearing headers are present, they carry a very small number of options, usually only one.

These assumptions suggest the following approach to laying out the fields of an option: order the fields from smallest to largest, with no interior padding, then derive the alignment requirement for the entire option based on the alignment requirement of the largest field (up to a maximum alignment of 8 octets). This approach is illustrated in the following examples:

Example 1

If an option X required two data fields, one of length 8 octets and one of length 4 octets, it would be laid out as follows:

```
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                | Option Type=X |Opt Data Len=12|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         8-octet field                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Its alignment requirement is 8n+2, to ensure that the 8-octet field starts at a multiple-of-8 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  | Hdr Ext Len=1 | Option Type=X |Opt Data Len=12|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         8-octet field                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Example 2

If an option Y required three data fields, one of length 4 octets, one of length 2 octets, and one of length 1 octet, it would be laid out as follows:

```
                                                +-+-+-+-+-+-+-+-+
                                                | Option Type=Y |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Opt Data Len=7 | 1-octet field |         2-octet field         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Its alignment requirement is 4n+3, to ensure that the 4-octet field starts at a multiple-of-4 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  | Hdr Ext Len=1 | Pad1 Option=0 | Option Type=Y |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Opt Data Len=7 | 1-octet field |         2-octet field         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PadN Option=1 |Opt Data Len=2 |       0       |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Example 3

A Hop-by-Hop or Destination Options header containing both options X and Y from Examples 1 and 2 would have one of the two following formats, depending on which option appeared first:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  | Hdr Ext Len=3 | Option Type=X |Opt Data Len=12|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         8-octet field                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PadN Option=1 |Opt Data Len=1 |       0       | Option Type=Y |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Opt Data Len=7 | 1-octet field |         2-octet field         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PadN Option=1 |Opt Data Len=2 |       0       |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  | Hdr Ext Len=3 | Pad1 Option=0 | Option Type=Y |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Opt Data Len=7 | 1-octet field |         2-octet field         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PadN Option=1 |Opt Data Len=4 |       0       |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       0       |       0       | Option Type=X |Opt Data Len=12|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         8-octet field                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Appendix B. Changes Since RFC2460

This memo has the following changes from RFC2460.

- o Removed IP Next Generation from the Abstract.

- o Added text in Section 1 that the Data Transmission Order is the same as IPv4 as defined in RFC791.

- o Clarified the text in Section 3 about decrementing the hop limit.

- o Clarification that extension headers (except for the hop-by-hop options header) are not processed, inserted, or deleted by any node along a packet's delivery path.

- o Changed requirement for the Hop-by-Hop Options header to a may, and added a note to indicate what is expected regarding the Hop-by-Hop Options header.

- o Added paragraph to Section 4 to clarify how Extension Headers are numbered and which are upper-layer headers.

- o Add reference to the end of Section 4 to IPv6 Extension Header IANA registry.

- o Incorporate the updates from RFC5095 and RFC5871 to remove the description of the RH0 Routing Header, that the allocations guidelines for routing headers are specified in RFC5871, and removed RH0 Routing Header from the list of required extension headers.

- o Revised Section 4.5 on IPv6 Fragmentation based on updates from RFC5722, RFC6946 RFC7112, and RFC8021. This include:

  - Revised the text to handle the case of fragments that are whole datagrams (i.e., both the Fragment Offset field and the M flag are zero). If received they should be processed as a reassembled packet. Any other fragments that match should be processed independently. The revised Fragment creation process was modified to not create whole datagram fragments (Fragment Offset field and the M flag are zero).

  - Changed the text to require that IPv6 nodes must not create overlapping fragments. Also, when reassembling an IPv6 datagram, if one or more its constituent fragments is determined to be an overlapping fragment, the entire datagram (and any constituent fragments) must be silently discarded. Includes a clarification that no ICMP error message should be sent if overlapping fragments are received.

  - Revised the text to require that all headers through the first Upper-Layer Header are in the first fragment. This changed the text describing how packets are fragmented and reassembled, and added a new error case.

  - Added text to Fragment Header process on handling exact duplicate fragments.

  - Updated the Fragmentation header text to correct the inclusion of AH and note no next header case.

  - Change terminology in Fragment header section from "Unfragmentable Headers" to "Per-Fragment Headers".

  - Removed the paragraph in Section 5 that required including a fragment header to outgoing packets if a ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280.

  - Changed the text to clarify MTU restriction and 8-byte restrictions, and noting the restriction on headers in first fragment.

- In Section 4.5 added clarification noting that some fields in the IPv6 header may also vary across the fragments being reassembled and that other specifications may provide additional instructions for how they should be reassembled. For example, Section 5.3 of [RFC3168].

- Incorporated the update from RFC6564 to add a new Section 4.8 that describes recommendations for defining new Extension headers and options.

- Added text to Section 5 to define "IPv6 minimum link MTU".

- Simplify the text in Section 6 about Flow Labels and remove Appendix A, and instead point to the current specifications of the IPv6 Flow Label field as defined in [RFC6437] and the Traffic Class as defined in [RFC2474] and [RFC3168].

- Incorporate the update in made by RFC6935 "UDP Checksums for Tunneled Packets" in Section 8. Added an exception to the default behaviour for the handling of handling UDP packets with zero checksums for tunnels.

- Add instruction to Section 9 "IANA Considerations" to change references to RFC2460 to this document

- Revised and expanded Section 10 "Security Considerations".

- Add a paragraph to the acknowledgement section acknowledging the authors of the updating documents

- Update references to current versions and assign references to normative and informative.

- Changes to resolve the open Errata on RFC2460. These are:

Errata ID: 2541: This errata notes that RFC2460 didn't update RFC2205 when the length of the Flow Label was changed from 24 to 20 bits from RFC1883. This issue was resolved in RFC6437 where the Flow Label is defined. This draft now references RFC6437. No change is required.

Errata ID: 4279: This errata noted that the specification doesn't handle the case of a forwarding node receiving a packet with a zero Hop Limit. This is fixed in Section 3 of this draft.

Errata ID: 2843: This errata is marked rejected. No change was made.

B.1. Change History Since RFC2460

NOTE TO RFC EDITOR: Please remove this subsection prior to RFC Publication

This section describes change history made in each Internet Draft that went into producing this version. The numbers identify the Internet-Draft version in which the change was made.

Working Group Internet Drafts

13) Added link to reference to RFC6564 in Section 4.8.

13) Added text to Section 5 to define "IPv6 minimum link MTU".

13) Editorial changes.

12) Editorial changes (remove old duplicate paragraph).

11) In Section 4.5 added clarification noting that some fields in the IPv6 header may also vary across the fragments being reassembled and that other specifications may provide additional instructions for how they should be reassembled. For example, Section 5.3 of [RFC3168].

11) In Section 4 restructured text including separated behaviors of extension headers and the hop-by-hop option header, removed "examine" from first paragraph about extension headers, and removed reference to RFC7045 because "examine" was removed (RFC7045 is referenced in Security Considerations). Also removed "including the source and

destination nodes" from paragraph about the hop-by-hop options header.

11) Revised Section 4.8 to make it closer to the update done by RFC6554 that updated it and reordered the paragraphs.

11) Reordered items in Appendix B "Changes Since RFC2460" to match the order of the document.

11) Editorial changes.

10) Revised and expanded Security Consideration Section based on IESG Discuss comments.

10) Editorial changes.

09) Based on results of IETF last call, changed text in Section 4 to add clarification that extension headers are not examined, processed, inserted, or deleted by any node along a packet's delivery path.

09) Changed reference from draft-ietf-6man-rfc4291bis to RFC4291 because the bis draft won't be advanced as the same time.

09) Revised "Changes since RFC2460" Section to have a summary of changes since RFC2460 and a separate subsection with a change history of each Internet Draft. This subsection will be removed when the RFC is published.

09) Editorial changes.

08) Revised header insertion text in Section 4 based on the results of w.g. survey that concluded to describe the problems with header insertion.

08) Editorial changes.

07) Expanded Security Considerations section to include both IPsec and encryption at higher levels in the protocol stack as ways to mitigate IP level security issues.

07) Added paragraph to Section 4 to clarify how Extension Headers are numbered and which are upper-layer headers.

07) Moved the text regarding network duplicated fragments to the received fragment error section.

07) Added clarification that no ICMP error message should be sent if overlapping fragments are received.

07) Revised the text in Section 4.8 regarding new hop-by-hop options and new Extension headers to be closer to the -05 version.

07) Added additional registries to the IANA Considerations section that IANA needs to update.

07) Editorial changes.

06) Added the Routing Header to the list required extension headers that a full implementation includes.

06) Moved the text in Section 4.5 regarding the handling of received overlapping fragments to the list of error conditions

06) Rewrote the text in Section 4.8 "Defining New Extension Headers and Options" to be clearer and remove redundant text.

06) Editorial changes.

05) Changed requirement for the Hop-by-Hop Options header from a should to a may, and added a note to indicate what is expected.

05) Corrected reference to point to draft-ietf-6man-rfc4291bis instead of draft-hinden-6man-rfc4291bis.

05) Change to text regarding not inserting extension headers to cite using encapsulation as an example.

04) Changed text discussing Fragment ID selection to refer to RFC7739 for example algorithms.

04) Editorial changes.

03) Clarified the text about decrementing the hop limit.

03) Removed IP Next Generation from the Abstract.

03) Add reference to the end of Section 4 to IPv6 Extension Header IANA registry.

03) Editorial changes.

02) Added text to Section 4.8 "Defining New Extension Headers and Options" clarifying why no new hop by hop extension headers should be defined.

02) Added text to Fragment Header process on handling exact duplicate fragments.

02) Editorial changes.

01) Added text that Extension headers must never be inserted by any node other than the source of the packet.

01) Change "must" to "should" in Section 4.3 on the Hop-by-Hop header.

01) Added text that the Data Transmission Order is the same as IPv4 as defined in RFC791.

01) Updated the Fragmentation header text to correct the inclusion of AH and note no next header case.

01) Change terminology in Fragment header section from "Unfragmentable Headers" to "Per-Fragment Headers".

01) Removed paragraph in Section 5 that required including a fragment header to outgoing packets if a ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280. This is based on the update in RFC8021.

01) Changed to Fragmentation Header section to clarify MTU restriction and 8-byte restrictions, and noting the restriction on headers in first fragment.

01) Editorial changes.

00) Add instruction to the IANA to change references to RFC2460 to this document

00) Add a paragraph to the acknowledgement section acknowledging the authors of the updating documents

00) Remove old paragraph in Section 4 that should have been removed when incorporating the update from RFC7045.

00) Editorial changes.

Individual Internet Drafts

07) Update references to current versions and assign references to normative and informative.

07) Editorial changes.

06) The purpose of this draft is to incorporate the updates dealing with Extension headers as defined in RFC6564, RFC7045, and RFC7112. The changes include:

RFC6564: Added new Section 4.8 that describe recommendations for defining new Extension headers and options

RFC7045: The changes were to add a reference to RFC7045, change the requirement for processing the hop-by-hop option to a should, and added a note that due to performance restrictions some nodes won’t process the Hop-by-Hop Option header.

RFC7112: The changes were to revise the Fragmentation Section (Section 4.5) to require that all headers through the first Upper-Layer Header are in the first fragment. This changed the text describing how packets are fragmented and reassembled and added a new error case.

06) Editorial changes.

05) The purpose of this draft is to incorporate the updates dealing with fragmentation as defined in RFC5722 and RFC6946. Note: The issue relating to the handling of exact duplicate fragments identified on the mailing list is left open.

05) Fix text in the end of Section 4 to correct the number of extension headers defined in this document.

05) Editorial changes.

04) The purpose of this draft is to update the document to incorporate the update made by RFC6935 "UDP Checksums for Tunneled Packets".

04) Remove Routing (Type 0) header from the list of required extension headers.

04) Editorial changes.

03) The purpose of this draft is to update the document for the deprecation of the RH0 Routing Header as specified in RFC5095 and the allocations guidelines for routing headers as specified in RFC5871. Both of these RFCs updated RFC2460.

02) The purpose of this version of the draft is to update the document to resolve the open Errata on RFC2460.

Errata ID: 2541: This errata notes that RFC2460 didn’t update RFC2205 when the length of the Flow Label was changed from 24 to 20 bits from RFC1883. This issue was resolved in RFC6437 where the Flow Label is defined. This draft now references RFC6437. No change is required.

Errata ID: 4279: This errata noted that the specification doesn’t handle the case of a forwarding node receiving a packet with a zero Hop Limit. This is fixed in Section 3 of this draft. Note: No change was made regarding host behaviour.

Errata ID: 2843: This errata is marked rejected. No change is required.

02) Editorial changes to the Flow Label and Traffic Class text.

01) The purpose of this version of the draft is to update the document to point to the current specifications of the IPv6 Flow Label field as defined in [RFC6437] and the Traffic Class as defined in [RFC2474] and [RFC3168].

00) The purpose of this version is to establish a baseline from RFC2460. The only intended changes are formatting (XML is slightly different from .nroff), differences between an RFC

and Internet Draft, fixing a few ID Nits, and updates to the authors information. There should not be any content changes to the specification.

Authors' Addresses

Stephen E. Deering
Retired
Vancouver, British Columbia
Canada

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, CA 94070
USA

Email: bob.hinden@gmail.com

Network Working Group
Internet-Draft
Obsoletes: 4291 (if approved)
Intended status: Standards Track
Expires: January 4, 2018

R. Hinden
Check Point Software
S. Deering
Retired

July 3, 2017

IP Version 6 Addressing Architecture
draft-ietf-6man-rfc4291bis-09

Abstract

This specification defines the addressing architecture of the IP Version 6 (IPv6) protocol. The document includes the IPv6 addressing model, text representations of IPv6 addresses, definition of IPv6 unicast addresses, anycast addresses, and multicast addresses, and an IPv6 node's required addresses.

This document obsoletes RFC 4291, "IP Version 6 Addressing Architecture".

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This specification defines the addressing architecture of the IP Version 6 protocol. It includes the basic formats for the various types of IPv6 addresses (unicast, anycast, and multicast).

2. IPv6 Addressing

IPv6 addresses are 128-bit identifiers for interfaces and sets of interfaces (where "interface" is as defined in Section 2 of [I-D.ietf-6man-rfc2460bis]). There are three types of addresses:

Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.

Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance).

Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

There are no broadcast addresses in IPv6, their function being superseded by multicast addresses.

In this document, fields in addresses are given a specific name, for example, "subnet". When this name is used with the term "ID" for identifier after the name (e.g., "subnet ID"), it refers to the contents of the named field. When it is used with the term "prefix" (e.g., "subnet prefix"), it refers to all of the address from the left up to and including this field.

Note: The term "prefix" is used in several different contexts for IPv6: a prefix used by a routing protocol, a prefix used by a node to determine if another node is connected to the same link, and a prefix used to construct the complete address of a node.

In IPv6, all zeros and all ones are legal values for any field, unless specifically excluded. Specifically, prefixes may contain, or end with, zero-valued fields.

2.1. Addressing Model

IPv6 addresses of all types are assigned to interfaces, not nodes. An IPv6 unicast address refers to a single interface. Since each interface belongs to a single node, any of that node's interfaces' unicast addresses may be used as an identifier for the node.

All interfaces are required to have at least one Link-Local unicast address (see Section 2.7 for additional required addresses). A single interface may also have multiple IPv6 addresses of any type (unicast, anycast, and multicast) or scope. Unicast addresses with a scope greater than link-scope are not needed for interfaces that are not used as the origin or destination of any IPv6 packets to or from non-neighbors. This is sometimes convenient for point-to-point interfaces. There is one exception to this addressing model:

> A unicast address or a set of unicast addresses may be assigned to multiple physical interfaces if the implementation treats the multiple physical interfaces as one interface when presenting it to the internet layer. This is useful for load-sharing over multiple physical interfaces.

Currently, IPv6 continues the IPv4 model in that a subnet prefix is associated with one link. Multiple subnet prefixes may be assigned to the same link. The relationship between links and IPv6 subnet prefixes differs from the IPv4 model in that all nodes automatically configure an address from the link-local prefix. A host is by definition on-link with it's default router, and that unicast addresses are not automatically associated with an on-link prefix. See [RFC5942] "The IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes" for more details.

2.2. Text Representation of IPv6 Addresses

2.2.1. Text Representation of Addresses

There are three conventional forms for representing IPv6 addresses as text strings:

1. The preferred form is x:x:x:x:x:x:x:x, where the 'x's are one to four hexadecimal digits of the eight 16-bit pieces of the address. Examples:

```
abcd:ef01:2345:6789:abcd:ef01:2345:6789
2001:db8:0:0:8:800:200c:417a
```

   Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.).

2. Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier, a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address.

   For example, the following addresses

```
2001:db8:0:0:8:800:200c:417a  a unicast address
ff01:0:0:0:0:0:0:101          a multicast address
0:0:0:0:0:0:0:1               the loopback address
0:0:0:0:0:0:0:0               the unspecified address
```

   may be represented as

```
2001:db8::8:800:200c:417a     a unicast address
ff01::101                     a multicast address
::1                           the loopback address
::                            the unspecified address
```

3. An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). Examples:

```
      0:0:0:0:0:0:13.1.68.3
      0:0:0:0:0:ffff:129.144.52.38
```

or in compressed form:

```
      ::13.1.68.3
      ::ffff:129.144.52.38
```

2.2.2. Text Representation of Address Prefixes

The text representation of IPv6 address prefixes is similar to the way IPv4 address prefixes are written in Classless Inter-Domain Routing (CIDR) notation [RFC4632]. An IPv6 address prefix is represented by the notation:

```
ipv6-address/prefix-length
```

where

ipv6-address is an IPv6 address in any of the notations listed in Section 2.2.

prefix-length is a decimal value specifying how many of the leftmost contiguous bits of the address comprise the prefix.

For example, the following are legal representations of the 60-bit prefix 20010db80000cd3 (hexadecimal):

```
2001:0db8:0000:cd30:0000:0000:0000:0000/60

2001:0db8::cd30:0:0:0:0/60

2001:0db8:0:cd30::/60
```

The following are NOT legal representations of the above prefix:

```
2001:0db8:0:cd3/60   may drop leading zeros, but not trailing
                     zeros, within any 16-bit chunk of the address

2001:0db8::cd30/60   address to left of "/" expands to
                     2001:0db8:0000:0000:0000:0000:0000:cd30

2001:0db8::cd3/60    address to left of "/" expands to
                     2001:0db8:0000:0000:0000:0000:0000:0cd3
```

When writing both a node address and a prefix of that node address (e.g., the node’s subnet prefix), the two can be combined as follows:

```
the node address        2001:0db8:0:cd30:123:4567:89ab:cdef
and its subnet prefix   2001:0db8:0:cd30::/60
```

```
can be abbreviated as   2001:0db8:0:cd30:123:4567:89ab:cdef/60
```

2.2.3. Recommendation for outputting IPv6 addresses

This section provides a recommendation for systems generating and outputting IPv6 addresses as text. Note, all implementations must accept and process all addresses in the formats defined in the previous two sections of this document. Background on this recommendation can be found in [RFC5952].

The recommendations are as follows:

1. The hexadecimal digits "a", "b", "c", "d", "e", and "f" in an IPv6 address must be represented in lowercase.

2. Leading zeros in a 16-Bit Field must be suppressed. For example,

   ```
   2001:0db8::0001
   ```

   is not correct and must be represented as

   ```
   2001:db8::1
   ```

3. A single 16-bit 0000 field must be represented as 0.

   The use of the symbol "::" must be used to its maximum capability. For example:

   ```
   2001:db8:0:0:0:0:2:1
   ```

   must be shortened to

    2001:db8::2:1

Likewise,

    2001:db8::0:1

is not correct, because the symbol "::" could have been used to produce a shorter representation

    2001:db8::1.

4. When there is an alternative choice in the placement of a "::", the longest run of consecutive 16-bit 0 fields must be shortened, that is, in

    2001:0:0:1:0:0:0:1

   the sequence with three consecutive zero fields is shortened to

    2001:0:0:1::1

5. When the length of the consecutive 16-bit 0 fields are equal, for example

    2001:db8:0:0:1:0:0:1

   the first sequence of zero bits must be shortened. For example

    2001:db8::1:0:0:1

   is the correct representation.

6. The symbol "::" must not be used to shorten just one 16-bit 0 field. For example, the representation

2001:db8:0:1:1:1:1:1

is correct, but

2001:db8::1:1:1:1:1

is not correct.

7. The text representation method describe in this section should also be use for text Representation of IPv6 Address Prefixes. For example

   2001:0db8:0000:cd30:0000:0000:0000:0000/60

   should be shown as

   2001:0db8:0:cd30::/60

8. The text representation method describe in this section should be applied for IPv6 addresses with embedded IPv4 address. For example

   0:0:0:0:0:ffff:192.0.2.1

   should be shown as

   ::ffff:192.0.2.1

2.3. Address Type Identification

The type of an IPv6 address is identified by the high-order bits of the address, as follows:

Address type	Binary prefix	IPv6 notation	Section
Unspecified	00...0 (128 bits)	::/128	2.4.2
Loopback	00...1 (128 bits)	::1/128	2.4.3
Multicast	11111111	ff00::/8	2.6
Link-Local unicast	1111111010	fe80::/10	2.4.6
Global Unicast	(everything else)		

Anycast addresses are taken from the unicast address spaces (of any scope) and are not syntactically distinguishable from unicast addresses.

The general format of Global Unicast addresses is described in Section 2.4.4. Some special-purpose subtypes of Global Unicast addresses that contain embedded IPv4 addresses (for the purposes of IPv4-IPv6 interoperation) are described in Section 2.4.5.

Future specifications may redefine one or more sub-ranges of the Global Unicast space for other purposes, but unless and until that happens, implementations must treat all addresses that do not start with any of the above-listed prefixes as Global Unicast addresses.

The current assigned IPv6 prefixes and references to their usage can be found in the IANA Internet Protocol Version 6 Address Space registry [IANA-AD] and the IANA IPv6 Special-Purpose Address Registry [IANA-SP].

2.4. Unicast Addresses

IPv6 unicast addresses are aggregatable with prefixes of arbitrary bit-length, similar to IPv4 addresses under Classless Inter-Domain Routing.

IPv6 unicast routing is based on prefixes of any valid length up to 128 [BCP198].

There are several types of unicast addresses in IPv6, in particular, Global Unicast, Local unicast, and Link-Local unicast. There are also some special-purpose subtypes of Global Unicast, such as IPv6 addresses with embedded IPv4 addresses. Additional address types or subtypes can be defined in the future.

IPv6 nodes may have considerable or little knowledge of the internal structure of the IPv6 address, depending on the role the node plays (for instance, host versus router). At a minimum, a node may consider that unicast addresses (including its own) have no internal structure:

```
|                          128 bits                              |
+----------------------------------------------------------------+
|                         node address                           |
+----------------------------------------------------------------+
```

A slightly more complex node may additionally be aware of subnet prefix(es) for the link(s) it is attached to, where different addresses may have different values for n:

```
|          n bits               |           128-n bits           |
+-------------------------------+--------------------------------+
|       subnet prefix           |           interface ID         |
+-------------------------------+--------------------------------+
```

Though a very simple router may have no knowledge of the internal structure of IPv6 unicast addresses, routers will more generally have knowledge of one or more of the hierarchical boundaries for the operation of routing protocols. The known boundaries will differ from router to router, depending on what positions the router holds in the routing hierarchy.

Except for the knowledge of the subnet boundary discussed in the previous paragraphs, nodes should not make any assumptions about the structure of an IPv6 address.

2.4.1. Interface Identifiers

Interface identifiers in IPv6 unicast addresses are used to identify interfaces on a link. They are required to be unique within a subnet prefix. It is recommended that the same interface identifier not be assigned to different nodes on a link. They may also be unique over a broader scope. The same interface identifier may be used on multiple interfaces on a single node, as long as they are attached to different subnets.

Interface IDs must be viewed outside of the node that created Interface ID as an opaque bit string without any internal structure.

Note that the uniqueness of interface identifiers is independent of the uniqueness of IPv6 addresses. For example, a Global Unicast address may be created with an interface identifier that is only unique on a single subnet, and a Link-Local address may be created with interface identifier that is unique over multiple subnets.

Interface Identifiers are 64 bit long except if the first three bits of the address are 000, or when the addresses are manually configured, or by exceptions defined in standards track documents. The rationale for using 64 bit Interface Identifiers can be found in

[RFC7421]. An example of a standards track exception is [RFC6164] that standardises 127 bit prefixes on inter-router point-to-point links.

The details of forming interface identifiers are defined in other specifications, such as "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" [RFC4941] or "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)"[RFC7217]. Specific cases are described in appropriate "IPv6 over <link>" specifications, such as "IPv6 over Ethernet" [RFC2464] and "Transmission of IPv6 Packets over ITU-T G.9959 Networks" [RFC7428]. The security and privacy considerations for IPv6 address generation is described in [RFC7721].

Earlier versions of this document described a method of forming interface identifiers derived from IEEE MAC-layer addresses call Modified EUI-64 format. These are described in Appendix A and are no longer recommended.

2.4.2. The Unspecified Address

The address 0:0:0:0:0:0:0:0 is called the unspecified address. It must never be assigned to any node. It indicates the absence of an address. One example of its use is in the Source Address field of any IPv6 packets sent by an initializing host before it has learned its own address.

The unspecified address must not be used as the destination address of IPv6 packets or in IPv6 Routing headers. An IPv6 packet with a source address of unspecified must never be forwarded by an IPv6 router.

2.4.3. The Loopback Address

The unicast address 0:0:0:0:0:0:0:1 is called the loopback address. It may be used by a node to send an IPv6 packet to itself. It must not be assigned to any physical interface. It is treated as having Link-Local scope, and may be thought of as the Link-Local unicast address of a virtual interface (typically called the "loopback interface") to an imaginary link that goes nowhere.

The loopback address must not be used as the source address in IPv6 packets that are sent outside of a single node. An IPv6 packet with a destination address of loopback must never be sent outside of a single node and must never be forwarded by an IPv6 router. A packet received on an interface with a destination address of loopback must be dropped.

2.4.4. Global Unicast Addresses

The general format for IPv6 Global Unicast addresses is as follows:

```
|         n bits         |   m bits  |       128-n-m bits         |
+------------------------+-----------+----------------------------+
| global routing prefix  | subnet ID |       interface ID         |
+------------------------+-----------+----------------------------+
```

where the global routing prefix is a (typically hierarchically-structured) value assigned to a site (a cluster of subnets/links), the subnet ID is an identifier of a link within the site, and the interface ID is as defined in Section 2.4.1.

Examples of Global Unicast addresses that start with binary 000 are the IPv6 address with embedded IPv4 addresses described in Section 2.4.5. An example of global addresses starting with a binary value other than 000 (and therefore having a 64-bit interface ID field) can be found in [RFC3587].

2.4.5. IPv6 Addresses with Embedded IPv4 Addresses

Two types of IPv6 addresses are defined that carry an IPv4 address in the low-order 32 bits of the address. These are the "IPv4-Compatible IPv6 address" and the "IPv4-mapped IPv6 address".

2.4.5.1. IPv4-Compatible IPv6 Address

The "IPv4-Compatible IPv6 address" was defined to assist in the IPv6 transition. The format of the "IPv4-Compatible IPv6 address" is as follows:

```
|                80 bits               | 16 |      32 bits        |
+--------------------------------------+--------------------------+
|0000..............................0000|0000|    IPv4 address     |
+--------------------------------------+----+---------------------+
```

Note: The IPv4 address used in the "IPv4-Compatible IPv6 address" must be a globally-unique IPv4 unicast address.

The "IPv4-Compatible IPv6 address" is now deprecated because the current IPv6 transition mechanisms no longer use these addresses. New or updated implementations are not required to support this address type.

2.4.5.2. IPv4-Mapped IPv6 Address

A second type of IPv6 address that holds an embedded IPv4 address is defined. This address type is used to represent the addresses of IPv4 nodes as IPv6 addresses. The format of the "IPv4-mapped IPv6 address" is as follows:

```
|                80 bits               | 16 |      32 bits        |
+--------------------------------------+--------------------------+
|0000..............................0000|ffff|    IPv4 address     |
+--------------------------------------+----+---------------------+
```

See [RFC4038] for background on the usage of the "IPv4-mapped IPv6 address".

2.4.6. Link-Local IPv6 Unicast Addresses

Link-Local addresses are for use on a single link. Link-Local addresses have the following format:

```
|   10     |
|  bits    |         54 bits         |          64 bits           |
+----------+-------------------------+----------------------------+
|1111111010|           0             |       interface ID         |
+----------+-------------------------+----------------------------+
```

Link-Local addresses are designed to be used for addressing on a single link for purposes such as automatic address configuration, neighbor discovery, or when no routers are present.

Routers must not forward any packets with Link-Local source or destination addresses to other links.

2.4.7. Other Local Unicast IPv6 Addresses

Unique Local Addresses (ULA) [RFC4193], the current form of Local IPv6 Addresses, are intended to be used for local communications, have global unicast scope, and are not expected to be routable on the global Internet.

Site-Local addresses, deprecated by [RFC3879], the previous form of Local IPv6 Addresses, were originally designed to be used for addressing inside of a site without the need for a global prefix.

The special behavior of Site-Local defined in [RFC3513] must no longer be supported in new implementations (i.e., new implementations must treat this prefix as Global Unicast). Existing implementations and deployments may continue to use this prefix.

2.5. Anycast Addresses

An IPv6 anycast address is an address that is assigned to more than one interface (typically belonging to different nodes), with the property that a packet sent to an anycast address is routed to the "nearest" interface having that address, according to the routing protocols’ measure of distance.

Anycast addresses are allocated from the unicast address space, using any of the defined unicast address formats. Thus, anycast addresses are syntactically indistinguishable from unicast addresses. When a unicast address is assigned to more than one interface, thus turning it into an anycast address, the nodes to which the address is assigned must be explicitly configured to know that it is an anycast address.

For any assigned anycast address, there is a longest prefix P of that address that identifies the topological region in which all interfaces belonging to that anycast address reside. Within the region identified by P, the anycast address must be maintained as a separate entry in the routing system (commonly referred to as a "host route"); outside the region identified by P, the anycast address may be aggregated into the routing entry for prefix P.

Note that in the worst case, the prefix P of an anycast set may be the null prefix, i.e., the members of the set may have no topological locality. In that case, the anycast address must be maintained as a separate routing entry throughout the entire Internet, which presents a severe scaling limit on how many such "global" anycast sets may be supported. Therefore, it is expected that support for global anycast sets may be unavailable or very restricted.

One expected use of anycast addresses is to identify the set of routers belonging to an organization providing Internet service. Such addresses could be used as intermediate addresses in an IPv6 Routing header, to cause a packet to be delivered via a particular service provider or sequence of service providers.

Some other possible uses are to identify the set of routers attached to a particular subnet, or the set of routers providing entry into a particular routing domain.

2.5.1. Required Anycast Address

The Subnet-Router anycast address is predefined. Its format is as follows:

```
|                        n bits                  |   128-n bits   |
+------------------------------------------------+----------------+
|                  subnet prefix                 | 00000000000000 |
+------------------------------------------------+----------------+
```

The "subnet prefix" in an anycast address is the prefix that identifies a specific link. This anycast address is syntactically the same as a unicast address for an interface on the link with the interface identifier set to zero.

Packets sent to the Subnet-Router anycast address will be delivered to one router on the subnet. All routers are required to support the Subnet-Router anycast addresses for the subnets to which they have interfaces.

The Subnet-Router anycast address is intended to be used for applications where a node needs to communicate with any one of the set of routers.

2.6. Multicast Addresses

An IPv6 multicast address is an identifier for a group of interfaces (typically on different nodes). An interface may belong to any number of multicast groups. Multicast addresses have the following format:

```
|   8    |  4 |  4 |                  112 bits                   |
+------- -+----+----+---------------------------------------------+
|11111111|flgs|scop|                  group ID                   |
+--------+----+----+---------------------------------------------+
```

binary 11111111 at the start of the address identifies the address as being a multicast address.

```
                                  +-+-+-+-+
flgs is a set of 4 flags:         |0|R|P|T|
                                  +-+-+-+-+
```

The high-order flag is reserved, and must be initialized to 0.

T = 0 indicates a permanently-assigned ("well-known") multicast address, assigned by the Internet Assigned Numbers Authority (IANA).

T = 1 indicates a non-permanently-assigned ("transient" or "dynamically" assigned) multicast address.

The P flag’s definition and usage can be found in [RFC3306].

The R flag’s definition and usage can be found in [RFC3956].

scop is a 4-bit multicast scope value used to limit the scope of the multicast group. The values are as follows:

0 reserved
1 Interface-Local scope
2 Link-Local scope
3 Realm-Local scope
4 Admin-Local scope
5 Site-Local scope
6 (unassigned)
7 (unassigned)
8 Organization-Local scope
9 (unassigned)
A (unassigned)
B (unassigned)
C (unassigned)
D (unassigned)
E Global scope
F reserved

Interface-Local scope spans only a single interface on a node and is useful only for loopback transmission of multicast. Packets with interface-local scope received from another node must be discarded.

Link-Local multicast scope spans the same topological region as the corresponding unicast scope.

Interface-Local, Link-Local, and Realm-Local scope boundaries are automatically derived from physical connectivity or other non-multicast-related configurations. Global scope has no boundary. The boundaries of all other non-reserved scopes of Admin-Local or larger are administratively configured. For reserved scopes, the way of configuring their boundaries will be defined when the semantics of the scope are defined.

According to [RFC4007], the zone of a Realm-Local scope must fall within zones of larger scope. Because the zone of a Realm-Local scope is configured automatically while the zones of larger scopes are configured manually, care must be taken in the definition of those larger scopes to ensure that the inclusion constraint is met.

Realm-Local scopes created by different network technologies are considered to be independent and will have different zone indices (see Section 6 of [RFC4007]). A router with interfaces on links using different network technologies does not forward traffic between the Realm-Local multicast scopes defined by those technologies.

Site-Local scope is intended to span a single site.

Organization-Local scope is intended to span multiple sites belonging to a single organization.

scopes labeled "(unassigned)" are available for administrators to define additional multicast regions.

group ID identifies the multicast group, either permanent or transient, within the given scope. Additional definitions of the multicast group ID field structure are provided in [RFC3306].

The "meaning" of a permanently-assigned multicast address is independent of the scope value. For example, if the "NTP servers group" is assigned a permanent multicast address with a group ID of 101 (hex), then

ff01:0:0:0:0:0:0:101 means all NTP servers on the same interface (i.e., the same node) as the sender.

ff02:0:0:0:0:0:0:101 means all NTP servers on the same link as the sender.

ff05:0:0:0:0:0:0:101 means all NTP servers in the same site as the sender.

ff0e:0:0:0:0:0:0:101 means all NTP servers in the Internet.

Non-permanently-assigned multicast addresses are meaningful only within a given scope. For example, a group identified by the non-permanent, site-local multicast address ff15:0:0:0:0:0:0:101 at one site bears no relationship to a group using the same address at a different site, nor to a non-permanent group using the same group ID with a different scope, nor to a permanent group with the same group ID.

Multicast addresses must not be used as source addresses in IPv6 packets or appear in any Routing header.

Routers must not forward any multicast packets beyond the scope indicated by the scop field in the destination multicast address.

Nodes must not originate a packet to a multicast address whose scop field contains the reserved value 0; if such a packet is received, it must be silently dropped. Nodes should not originate a packet to a multicast address whose scop field contains the reserved value F; if such a packet is sent or received, it must be treated the same as packets destined to a global (scop E) multicast address.

2.6.1. Pre-Defined Multicast Addresses

The following well-known multicast addresses are pre-defined. The group IDs defined in this section are defined for explicit scope values.

Use of these group IDs for any other scope values, with the T flag equal to 0, is not allowed.

```
   Reserved Multicast Addresses: ff00:0:0:0:0:0:0:0
                                 ff01:0:0:0:0:0:0:0
                                 ff02:0:0:0:0:0:0:0
                                 ff03:0:0:0:0:0:0:0
                                 ff04:0:0:0:0:0:0:0
                                 ff05:0:0:0:0:0:0:0
                                 ff06:0:0:0:0:0:0:0
                                 ff07:0:0:0:0:0:0:0
                                 ff08:0:0:0:0:0:0:0
                                 ff09:0:0:0:0:0:0:0
                                 ff0a:0:0:0:0:0:0:0
                                 ff0b:0:0:0:0:0:0:0
                                 ff0c:0:0:0:0:0:0:0
                                 ff0d:0:0:0:0:0:0:0
                                 ff0e:0:0:0:0:0:0:0
                                 ff0f:0:0:0:0:0:0:0
```

The above multicast addresses are reserved and shall never be assigned to any multicast group.

```
   All Nodes Addresses:          ff01:0:0:0:0:0:0:1
                                 ff02:0:0:0:0:0:0:1
```

The above multicast addresses identify the group of all IPv6 nodes, within scope 1 (interface-local) or 2 (link-local).

```
      All Routers Addresses:        ff01:0:0:0:0:0:0:2
                                    ff02:0:0:0:0:0:0:2
                                    ff05:0:0:0:0:0:0:2
```

The above multicast addresses identify the group of all IPv6 routers, within scope 1 (interface-local), 2 (link-local), or 5 (site-local).

```
      Solicited-Node Address:       ff02:0:0:0:0:1:ffxx:xxxx
```

Solicited-Node multicast address are computed as a function of a node’s unicast and anycast addresses. A Solicited-Node multicast address is formed by taking the low-order 24 bits of an address (unicast or anycast) and appending those bits to the prefix FF02:0:0:0:0:1:FF00::/104 resulting in a multicast address in the range

```
      ff02:0:0:0:0:1:ff00:0000
```

to

```
      ff02:0:0:0:0:1:ffff:ffff
```

For example, the Solicited-Node multicast address corresponding to the IPv6 address 4037::01:800:200e:8c6c is ff02::1:ff0e:8c6c. IPv6 addresses that differ only in the high-order bits (e.g., due to multiple high-order prefixes associated with different aggregations) will map to the same Solicited-Node address, thereby reducing the number of multicast addresses a node must join.

A node is required to compute and join (on the appropriate interface) the associated Solicited-Node multicast addresses for all unicast and anycast addresses that have been configured for the node’s interfaces (manually or automatically).

Additional defined multicast address can be found in the IANA IPv6 Multicast Address Allocation registry [IANA-MC]

2.7. A Node’s Required Addresses

A host is required to recognize the following addresses as identifying itself:

- o Its required Link-Local address for each interface.

- Any additional Unicast and Anycast addresses that have been configured for the node's interfaces (manually or automatically).

- The loopback address.

- The All-Nodes multicast addresses defined in Section 2.6.1.

- The Solicited-Node multicast address for each of its unicast and anycast addresses.

- Multicast addresses of all other groups to which the node belongs.

A router is required to recognize all addresses that a host is required to recognize, plus the following addresses as identifying itself:

- The Subnet-Router Anycast addresses for all interfaces for which it is configured to act as a router.

- All other Anycast addresses with which the router has been configured.

- The All-Routers multicast addresses defined in Section 2.6.1.

3. IANA Considerations

RFC4291 is referenced in a number of IANA registries. These include:

- Internet Protocol Version 6 Address Space [IANA-AD]

- IPv6 Global Unicast Address Assignments [IANA-GU]

- IPv6 Multicast Address Space Registry [IANA-MC]

- Application for an IPv6 Multicast Address [IANA-MA]

- Internet Protocol Version 6 (IPv6) Anycast Addresses [IANA-AC]

- IANA IPv6 Special-Purpose Address Registry [IANA-SP]

- Reserved IPv6 Interface Identifiers [IANA-ID]

- o Number Resources [IANA-NR]
- o Protocol Registries [IANA-PR]
- o Technical requirements for authoritative name servers [IANA-NS]
- o IP Flow Information Export (IPFIX) Entities [IANA-FE]

The IANA should update these references to point to this document.

There are also other references in IANA procedures documents that the IANA should investigate to see if they should be updated.

4. Security Considerations

IPv6 addressing documents do not have any direct impact on Internet infrastructure security. Authentication of IPv6 packets is defined in [RFC4302].

One area relavant to IPv6 addressing is privacy. IPv6 addresses can be created using interface identifiers constructed with unique stable tokens. The addresses created in this manner can be used to track the movement of devices across the Internet. Since earlier versions of this document were published, several approaches have been developed that mitigate these problems. These are described in "Security and Privacy Considerations for IPv6 Address Generation Mechanisms" [RFC7721], "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" [RFC4941], and "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)" [RFC7217].

5. Acknowledgments

The authors would like to acknowledge the contributions of Bill Simpson, Bob Fink, Bob Gilligan, Brian Carpenter, Christian Huitema, Deborah Estrin, Dimitry Haskin, Erik Nordmark, Greg Minshall, James Woodyatt. Jim Bound, Jun-ichiro Itojun Hagino, Larry Masinter, Mahmood Ali, Markku Savela, Matt Crawford, Paul Francis, Peter Ford, Roger Fajman, Scott Bradner, Sue Thomson, Suresh Krishnan, Tatuya Jinmei, Thomas Narten, Tom Harsch, Tony Li, and Yakov Rekhter.

The authors would also like to acknowledge the authors of the updating RFCs that were incorporated in this version of the document to move IPv6 to Internet Standard. This includes Marcelo Bagnulo, Congxiao Bao, Mohamed Boucadair, Brian Carpenter, Ralph Droms, Christian Huitema, Sheng Jiang, Seiichi Kawamura, Masanobu Kawashima, Xing Li, and Stig Venaas.

6. References

6.1. Normative References

[I-D.ietf-6man-rfc2460bis]
Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", draft-ietf-6man-rfc2460bis-13 (work in progress), May 2017.

6.2. Informative References

[BCP198] Boucadair, M., Petrescu, A., and F. Baker, "IPv6 Prefix Length Recommendation for Forwarding", BCP 198, RFC 7608, DOI 10.17487/RFC7608, July 2015, <http://www.rfc-editor.org/info/rfc7608>.

[EUI64] "IEEE, "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority"", March 1997, <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>.

[IANA-AC] "Internet Protocol Version 6 (IPv6) Anycast Addresses", <http://www.iana.org/assignments/ipv6-anycast-addresses/ipv6-anycast-addresses.xhtml>.

[IANA-AD] "Internet Protocol Version 6 Address Space", <https://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml>.

[IANA-FE] "IP Flow Information Export (IPFIX) Entities", <http://www.iana.org/assignments/ipfix/ipfix.xhtml>.

[IANA-GU] "IPv6 Global Unicast Address Assignments", <http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml>.

[IANA-ID] "IANA IPv6 Special-Purpose Address Registry", <http://www.iana.org/assignments/ipv6-interface-ids/ipv6-interface-ids.xhtml>.

[IANA-MA] "Application for an IPv6 Multicast Address", <https://www.iana.org/form/multicast-ipv6>.

[IANA-MC] "IPv6 Multicast Address Space Registry", <http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml>.

[IANA-NR] "Number Resources", <http://https://www.iana.org/numbers>.

[IANA-NS] "Technical requirements for authoritative name servers", <https://www.iana.org/help/nameserver-requirements>.

[IANA-PR] "Protocol Registries", <https://www.iana.org/protocols>.

[IANA-SP] "IANA IPv6 Special-Purpose Address Registry", <https://www.iana.org/assignments/iana-ipv6-special-registry/iana-ipv6-special-registry.xhtml>.

[RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <http://www.rfc-editor.org/info/rfc2464>.

[RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, DOI 10.17487/RFC3306, August 2002, <http://www.rfc-editor.org/info/rfc3306>.

[RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, DOI 10.17487/RFC3513, April 2003, <http://www.rfc-editor.org/info/rfc3513>.

[RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, DOI 10.17487/RFC3587, August 2003, <http://www.rfc-editor.org/info/rfc3587>.

[RFC3879] Huitema, C. and B. Carpenter, "Deprecating Site Local Addresses", RFC 3879, DOI 10.17487/RFC3879, September 2004, <http://www.rfc-editor.org/info/rfc3879>.

[RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", RFC 3956, DOI 10.17487/RFC3956, November 2004, <http://www.rfc-editor.org/info/rfc3956>.

[RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005, <http://www.rfc-editor.org/info/rfc4007>.

[RFC4038] Shin, M-K., Ed., Hong, Y-G., Hagino, J., Savola, P., and E. Castro, "Application Aspects of IPv6 Transition", RFC 4038, DOI 10.17487/RFC4038, March 2005, <http://www.rfc-editor.org/info/rfc4038>.

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

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

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <http://www.rfc-editor.org/info/rfc4632>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC5942] Singh, H., Beebee, W., and E. Nordmark, "IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes", RFC 5942, DOI 10.17487/RFC5942, July 2010, <http://www.rfc-editor.org/info/rfc5942>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <http://www.rfc-editor.org/info/rfc5952>.

[RFC6164] Kohno, M., Nitzan, B., Bush, R., Matsuzaki, Y., Colitti, L., and T. Narten, "Using 127-Bit IPv6 Prefixes on Inter-Router Links", RFC 6164, DOI 10.17487/RFC6164, April 2011, <http://www.rfc-editor.org/info/rfc6164>.

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

[RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <http://www.rfc-editor.org/info/rfc7421>.

[RFC7428] Brandt, A. and J. Buron, "Transmission of IPv6 Packets over ITU-T G.9959 Networks", RFC 7428, DOI 10.17487/RFC7428, February 2015, <http://www.rfc-editor.org/info/rfc7428>.

[RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <http://www.rfc-editor.org/info/rfc7721>.

Appendix A. Modified EUI-64 Format Interface Identifiers

Modified EUI-64 format-based interface identifiers may have universal scope when derived from a universal token (e.g., IEEE 802 48-bit MAC or IEEE EUI-64 identifiers [EUI64]) or may have local scope where a global token is not being used (e.g., serial links, tunnel end-points) or where global tokens are undesirable (e.g., temporary tokens for privacy [RFC4941].

Modified EUI-64 format interface identifiers are formed by inverting the "u" bit (universal/local bit in IEEE EUI-64 terminology) when forming the interface identifier from IEEE EUI-64 identifiers. In the resulting Modified EUI-64 format, the "u" bit is set to one (1) to indicate universal scope, and it is set to zero (0) to indicate local scope. The first three octets in binary of an IEEE EUI-64 identifier are as follows:

```
 0       0 0       1 1       2
|0       7 8       5 6       3|
+----+----+----+----+----+----+
|cccc|ccug|cccc|cccc|cccc|cccc|
+----+----+----+----+----+----+
```

written in Internet standard bit-order, where "u" is the universal/ local bit, "g" is the individual/group bit, and "c" is the bits of the company_id. Appendix A, "Creating Modified EUI-64 Format Interface Identifiers", provides examples on the creation of Modified EUI-64 format-based interface identifiers.

The motivation for inverting the "u" bit when forming an interface identifier is to make it easy for system administrators to hand configure non-global identifiers when hardware tokens are not available. This is expected to be the case for serial links and tunnel end-points, for example. The alternative would have been for these to be of the form 0200:0:0:1, 0200:0:0:2, etc., instead of the much simpler 0:0:0:1, 0:0:0:2, etc.

IPv6 nodes are not required to validate that interface identifiers created with modified EUI-64 tokens with the "u" bit set to universal are unique.

A.1. Creating Modified EUI-64 Format Interface Identifiers

Depending on the characteristics of a specific link or node, there are a number of approaches for creating Modified EUI-64 format interface identifiers. This appendix describes some of these approaches.

Links or Nodes with IEEE EUI-64 Identifiers

The only change needed to transform an IEEE EUI-64 identifier to an interface identifier is to invert the "u" (universal/local) bit. An example is a globally unique IEEE EUI-64 identifier of the form:

```
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|cccccc0gcccccccc|ccccccccmmmmmmmm|mmmmmmmmmmmmmmmm|mmmmmmmmmmmmmmmm|
+----------------+----------------+----------------+----------------+
```

where "c" is the bits of the assigned company_id, "0" is the value of the universal/local bit to indicate universal scope, "g" is individual/group bit, and "m" is the bits of the manufacturer-selected extension identifier. The IPv6 interface identifier would be of the form:

```
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|cccccc1gcccccccc|ccccccccmmmmmmmm|mmmmmmmmmmmmmmmm|mmmmmmmmmmmmmmmm|
+----------------+----------------+----------------+----------------+
```

The only change is inverting the value of the universal/local bit.

Links or Nodes with IEEE 802 48-bit MACs

[EUI64] defines a method to create an IEEE EUI-64 identifier from an IEEE 48-bit MAC identifier. This is to insert two octets, with hexadecimal values of 0xFF and 0xFE (see the Note at the end of appendix), in the middle of the 48-bit MAC (between the company_id and vendor-supplied id). An example is the 48-bit IEEE MAC with Global scope:

```
|0              1|1              3|3              4|
|0              5|6              1|2              7|
+----------------+----------------+----------------+
|cccccc0gcccccccc|ccccccccmmmmmmmm|mmmmmmmmmmmmmmmm|
+----------------+----------------+----------------+
```

where "c" is the bits of the assigned company_id, "0" is the value of the universal/local bit to indicate Global scope, "g" is individual/ group bit, and "m" is the bits of the manufacturer- selected extension identifier. The interface identifier would be of the form:

```
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|cccccc1gcccccccc|cccccccc11111111|11111110mmmmmmmm|mmmmmmmmmmmmmmmm|
+----------------+----------------+----------------+----------------+
```

When IEEE 802 48-bit MAC addresses are available (on an interface or a node), an implementation may use them to create interface identifiers due to their availability and uniqueness properties.

Links with Other Kinds of Identifiers

There are a number of types of links that have link-layer interface identifiers other than IEEE EUI-64 or IEEE 802 48-bit MACs. Examples include LocalTalk and Arcnet. The method to create a Modified EUI-64 format identifier is to take the link identifier (e.g., the LocalTalk 8-bit node identifier) and zero fill it to the left. For example, a LocalTalk 8-bit node identifier of hexadecimal value 0x4F results in the following interface identifier:

```
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|0000000000000000|0000000000000000|0000000000000000|0000000001001111|
+----------------+----------------+----------------+----------------+
```

Note that this results in the universal/local bit set to "0" to indicate local scope.

Links without Identifiers

There are a number of links that do not have any type of built-in identifier. The most common of these are serial links and configured tunnels. Interface identifiers that are unique within a subnet prefix must be chosen.

When no built-in identifier is available on a link, the preferred approach is to use a universal interface identifier from another interface or one that is assigned to the node itself. When using this approach, no other interface connecting the same node to the same subnet prefix may use the same identifier.

If there is no universal interface identifier available for use on the link, the implementation needs to create a local-scope interface identifier. The only requirement is that it be unique within a subnet prefix. There are many possible approaches to select a subnet-prefix-unique interface identifier. These include the following:

Manual Configuration
Node Serial Number
Other Node-Specific Token

The subnet-prefix-unique interface identifier should be generated in a manner such that it does not change after a reboot of a node or if interfaces are added or deleted from the node.

The selection of the appropriate algorithm is link and implementation dependent. The details on forming interface identifiers are defined in the appropriate "IPv6 over <link>" specification. It is strongly recommended that a collision detection algorithm be implemented as part of any automatic algorithm.

Note: [EUI64] actually defines 0xFF and 0xFF as the bits to be inserted to create an IEEE EUI-64 identifier from an IEEE MAC-48 identifier. The 0xFF and 0xFE values are used when starting with an IEEE EUI-48 identifier. The incorrect value was used in earlier versions of the specification due to a misunderstanding about the differences between IEEE MAC-48 and EUI-48 identifiers.

This document purposely continues the use of 0xFF and 0xFE because it meets the requirements for IPv6 interface identifiers (i.e., that they must be unique on the link), IEEE EUI-48 and MAC-48 identifiers are syntactically equivalent, and that it doesn’t cause any problems in practice.

Appendix B. CHANGES SINCE RFC 4291

This document has the following changes from RFC4291, "IP Version 6 Addressing Architecture":

o Added Note: to Section 2 that the term "prefix" is used in different contexts in IPv6: a prefix used by a routing protocol, a prefix used by a node to determine if another node is connected to the same link, and a prefix used to construct the complete address of a node.

o Added text to the last paragraph in Section 2.1 to clarify the differences on how subnets are hangled in IPv4 and IPv6, includes a reference to RFC5942 "The IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes".

o Incorporate the updates made by RFC5952 in Section 2.2.3 regarding the text format when outputting IPv6 addresses. A new section was added for this and addresses shown in this document were changed to lower case. This includes a reference to RFC5952.

o Incorporate the updates made by RFC6052. The change was to add a text in Section 2.3 that points to the IANA registries that records the prefix defined in RFC6052 and a number of other special use prefixes.

o Clarified text that 64 bit Interface IDs are used except when the first three bits of the address are 000, or addresses are manually configured, or when defined by a standard track document. Added text that Modified EUI-64 identifiers not recommended and moved the text describing the format to Appendix A. This text was moved from Section 2.4 and is now consolidated in Section 2.4.1. Also removed text in Section 2.4.4 relating to 64 bit Interface IDs.

o Added text to Section 2.4 summarizing IPv6 unicast routing and referencing BCP198, citing RFC6164 as an example of longer prefixes, and that IIDs are required to be 64 bits long as described in RFC7421.

o Incorporate the updates made by RFC7136 to deprecate the U and G bits in Modified EUI-64 format Internet IDs.

o Rename Section 2.4.7 to "Other Local Unicast Addresses" and rewrote the text to point to ULAs and say that Site-Local addresses were deprecated by RFC3879. The format of Site-Local was removed.

o Incorporate the updates made by RFC7346. The change was to add Realm-Local scope to the multicast scope table in Section 2.6, and add the updating text to the same section.

o Added a reference to the IANA Multicast address registry in Section 2.6.1.

o Added instructions in IANA Considerations to update references in the IANA registries that currently point to RFC4291 to point to this document.

o Expanded Security Considerations Section to discuss privacy issues related to using stable interface identifiers to create IPv6 addresses, and reference solutions that mitigate these issues such as RFC7721, RFC4941, RFC7271.

- o Add note to Section 5 section acknowledging the authors of the updating documents.

- o Updates to resolve the open Errata on RFC4291. These are:

    Errata ID: 3480: Corrects the definition of Interface-Local multicast scope to also state that packets with interface-local scope received from another node must be discarded.

    Errata ID: 1627: Remove extraneous "of" in Section 2.7.

    Errata ID: 2702: This errata is marked rejected. No change is required.

    Errata ID: 2735: This errata is marked rejected. No change is required.

    Errata ID: 4406: This errata is marked rejected. No change is required.

    Errata ID: 2406: This errata is marked rejected. No change is required.

    Errata ID: 863: This errata is marked rejected. No change is required.

    Errata ID: 864: This errata is marked rejected. No change is required.

    Errata ID: 866: This errata is marked rejected. No change is required.

- o Editorial changes.

B.1. Change History Since RFC4291

NOTE TO RFC EDITOR: Please remove this subsection prior to RFC Publication

This section describes change history made in each Internet Draft that went into producing this version. The numbers identify the Internet-Draft version in which the change was made.

Working Group Internet Drafts

09) Added text to the last paragraph in Section 2.1 to clarify the differences on how subnets are hangled in IPv4 and IPv6, includes a reference to RFC5942 "The IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes".

09) Removed short paragraph about manual configuration in Section 2.4.1 that was added in the -08 version.

09) Revised "Changes since RFC4291" Section to have a summary of changes since RFC4291 and a separate subsection with a change history of each Internet Draft. This subsection will be removed when the RFC is published.

09) Editorial changes.

08) Added Note: to Section 2 that the term "prefix" is used in different contexts in IPv6: a prefix used by a routing protocol, a prefix used by a node to determine if another node is connected to the same link, and a prefix used to construct the complete address of a node.

08) Based on results of IETF last call and extensive w.g. list discussion, revised text to clarify that 64 bit Interface IDs are used except when the first three bits of the address are 000, or addresses are manually configured, or when defined by a standard track document. This text was moved from Section 2.4 and is now consolidated in Section 2.4.1 Also removed text in Section 2.4.4 relating to 64 bit Interface IDs.

08) Removed instruction to IANA fix error in Port Number assignment. IANA fixed the error on 4 March 2017.

08) Editorial changes.

07) Added text to Section 2.4 summarizing IPv6 unicast routing and referencing BCP198, citing RFC6164 as an example of longer prefixes, and that IIDs are required to be 64 bits long as described in RFC7421.

07) Based on review by Brian Haberman added reference to RFC5952 in Section 2.2.3, corrected case errors in Section 2.6.1, and added a reference to the IANA Multicast address registry in Section 2.6.1.

07) Corrected errors in Section 2.2.3 where the examples in 7. and 8. were reversed.

07) Editorial changes.

06) Editorial changes.

05) Expanded Security Considerations Section to discuss privacy issues related to using stable interface identifiers to create IPv6 addresses, and reference solutions that mitigate these issues such as RFC7721, RFC4941, RFC7271.

05) Added instructions in IANA Considerations to update references in the IANA registries that currently point to RFC4291 to point to this document.

05) Rename Section 2.4.7 to "Other Local Unicast Addresses" and rewrote the text to point to ULAs and say that Site-Local addresses were deprecated by RFC3879. The format of Site-Local was removed.

05) Added to Section 2.4.1 a reference to RFC7421 regarding the background on the 64 bit boundary in Interface Identifiers.

05) Editorial changes.

04) Added text and a pointer to the ULA specification in Section 2.4.7

04) Removed old IANA Considerations text, this was left from the baseline text from RFC4291 and should have been removed earlier.

04) Editorial changes.

03) Changes references in Section 2.4.1 that describes the details of forming IIDs to RFC7271 and RFC7721.

02) Remove changes made by RFC7371 because there isn’t any known implementation experience.

01) Revised Section 2.4.1 on Interface Identifiers to reflect current approach, this included saying Modified EUI-64 identifiers not recommended and moved the text describing the format to Appendix A.

01) Editorial changes.

00) Working Group Draft.

00) Editorial changes.

Individual Internet Drafts

06) Incorporate the updates made by RFC7371. The changes were to the flag bits and their definitions in Section 2.6.

05) Incorporate the updates made by RFC7346. The change was to add Realm-Local scope to the multicast scope table in Section 2.6, and add the updating text to the same section.

04) Incorporate the updates made by RFC6052. The change was to add a text in Section 2.3 that points to the IANA registries that records the prefix defined in RFC6052 and a number of other special use prefixes.

03) Incorporate the updates made by RFC7136 to deprecate the U and G bits in Modified EUI-64 format Internet IDs.

03) Add note to the reference section acknowledging the authors of the updating documents.

03) Editorial changes.

02) Updates to resolve the open Errata on RFC4291. These are:

Errata ID: 3480: Corrects the definition of Interface-Local multicast scope to also state that packets with interface-local scope received from another node must be discarded.

Errata ID: 1627: Remove extraneous "of" in Section 2.7.

Errata ID: 2702: This errata is marked rejected. No change is required.

Errata ID: 2735: This errata is marked rejected. No change is required.

Errata ID: 4406: This errata is marked rejected. No change is required.

Errata ID: 2406: This errata is marked rejected. No change is required.

Errata ID: 863: This errata is marked rejected. No change is required.

Errata ID: 864: This errata is marked rejected. No change is required.

Errata ID: 866: This errata is marked rejected. No change is required.

02) Update references to current versions.

02) Editorial changes.

01) Incorporate the updates made by RFC5952 regarding the text format when outputting IPv6 addresses. A new section was added for this and addresses shown in this document were changed to lower case.

01) Revise this Section to document to show the changes from RFC4291.

01) Editorial changes.

00) Establish a baseline from RFC4291. The only intended changes are formatting (XML is slightly different from .nroff), differences between an RFC and Internet Draft, fixing a few ID Nits, and updates to the authors information. There should not be any content changes to the specification.

Authors' Addresses

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, CA 94070
USA

Email: bob.hinden@gmail.com

Stephen E. Deering
Retired
Vancouver, British Columbia
Canada