Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

J. Dong
Z. Li
Huawei Technologies
C. Xie
C. Ma
China Telecom

November 2, 2020

Carrying Virtual Transport Network Identifier in IPv6 Extension Header

draft-dong-6man-enhanced-vpn-vtn-id-02

Abstract

A Virtual Transport Network (VTN) is a virtual network which has a customized network topology and a set of dedicated or shared network resources allocated from the network infrastructure. A VTN can be used as the underlay for one or a group of VPNs to provide enhanced VPN (VPN+) services. In packet forwarding, some fields in data packet needs to be used to identify the VTN the packet belongs to, so that the VTN-specific processing can be performed.

This document proposes a new option type to carry VTN ID in an IPv6 extension headers to identify the Virtual Transport Network (VTN) the packet belongs to. The procedure for processing of the VTN option is also specified.

Status of This Memo

Table of Contents

1. Introduction

Virtual Private Networks (VPNs) provide different groups of users with logically isolated connectivity over a common shared network infrastructure. With the introduction of 5G, new service types may require connectivity services with advanced characteristics comparing to traditional VPNs, such as strict isolation from other services or guaranteed performance. These services are refered to as "enhanced VPNs" (VPN+). [I-D.ietf-teas-enhanced-vpn] describes a framework and candidate component technologies for providing VPN+ services.

The enhanced properties of VPN+ require tighter coordination and integration between the underlay network resources and the overlay network. VPN+ service can be built on a Virtual Transport Network (VTN) which has a customized network topology and a set of dedicated or shared network resources allocated from the underlay network. The overlay VPN together with the corresponding VTN in the underlay provide the VPN+ service. In the network, traffic of different VPN+ services need to be processed separately based on the topology and the network resources associated with the corresponding VTN.

[I-D.dong-teas-enhanced-vpn-vtn-scalability] describes the scalability considerations for VPN+, one of which is to improve the data plane scalability through the introduction of a dedicated identifier in data packets that is used to identify the VTN the packets belong to, so that VTN-specific packet processing can be performed. This is called Resource Independent (RI) VTN.

This document proposes a mechanism to carry the VTN ID in an IPv6 extension header [RFC8200] of a packet, so that the packet will be processed by network nodes using the network resources allocated to the corresponding VTN. The procedure for processing the VTN ID is also specified. This provides a scalable solution for enhanced VPN data plane, so that it may be used to support a large number of VTNs in an IPv6 network.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. New IPv6 Extension Header Option for VTN

A new option type "VTN" is defined to carry the Virtual Transport Network Identifier (VTN ID) in an IPv6 packet header. Its format is shown as below:

```
 Option   Option       Option
  Type   Data Len       Data
+--------+--------+--------+--------+
|BBCTTTTT|00000100|  4-octet VTN ID |
+--------+--------+--------+--------+
 Figure 1. The format of VTN Option
```

Option Type: 8-bit identifier of the type of option. The type of VTN option is to be assigned by IANA. The highest-order bits of the type field are defined as below:

- o BB 00 The highest-order 2 bits are set to 00 to indicate that a node which does not recognize this type will skip over it and continue processing the header.

- o C 0 The third highest-order bit are set to 0 to indicate this option does not change en route.

Opt Data Len: 8-bit unsigned integer indicates the length of the option Data field of this option, in octets. The value of Opt Data Len of VTN option SHOULD be set to 4.

Option Data: 4-octet identifier which uniquely identifies a VTN.

Editor’s note: The length of the VTN ID is defined as 4-octet for the matching with the 4-octet Single Network Slice Selection Assistance Information (S-NSSAI) defined in 3GPP [TS23501].

```
     8-bit              24-bit
+------------+-------------------------+
|    SST     |  Slice Differentiator   |
+------------+-------------------------+
```

Figure 2. The format of S-NSSAI

4. Procedures

As the VTN option needs to be processed by each node along the path for VTN-specific forwarding, it SHOULD be carried in IPv6 Hop-by-Hop options header when the Hop-by-Hop options header can be processed in forwarding plane by all the nodes along the path.

4.1. VTN Option Insertion

When an ingress node of an IPv6 domain receives a packet, according to traffic classification or mapping policy, the packet is steered into one of the VTNs in the network, then packet SHOULD be encapsulated in an outer IPv6 header, and the VTN-ID of the VTN which the packet is mapped to SHOULD be carried in the Hop-by-Hop options header associated with the outer IPv6 header.

4.2. VTN based Packet Forwarding

On receipt of a packet with the VTN option, each network node which can parse the VTN option SHOULD use the VTN ID to identify the VTN the packet belongs to. This means the forwarding behavior is based on both the destination IP address and the VTN option. The destination IP address is used for the lookup of the next-hop node, and VTN-ID can be used to determine the set of network resources reserved for processing and sending the packet to the next-hop node.

The egress node of the IPv6 domain SHOULD decapsulate the outer IPv6 header.

There can be different implementations for reserving local network resources to the VTNs. For example, on one interface, a subset of forwarding plane resource allocated to a particular VTN can be considered as a virtual sub-interface with dedicated bandwidth and other associated resources. In packet forwarding, the IPv6 destination address of the received packet is used to identify the next-hop and the outgoing interface, and the VTN ID is used to further identify the virtual sub-interface which is associated with the VTN on the outgoing interface.

Routers which do not support Hop-by-Hop options header SHOULD ignore the Hop-by-Hop options header and forward the packet only based on the destination IP address. Routers which support Hop-by-Hop Options header, but do not support the VTN option SHOULD ignore the Hop-by-Hop option and continue to forward the packet only based on the destination IP address.

5. Operational Considerations

As described in [RFC8200], nodes may be configured to ignore the Hop-by-Hop Options header, and in some implementations a packet containing a Hop-by-Hop Options header may be dropped or assigned to a slow processing path. This needs to be taken into consideration when VTN option is introduced to a network. The operator needs to make sure that all the network nodes in a VTN can either process Hop-by-Hop Options header in packet forwarding, or ignore the Hop-by-Hop Option header. In other word, packets steered into a VTN MUST NOT be dropped due to the existence of the Hop-by-Hop Options header. It is RECOMMENDED to configure all the nodes in a VTN to process the Hop-by-Hop Options header if there is a nob for this.

6. IANA Considerations

This document requests IANA to assign a new option type from "Destination Options and Hop-by-Hop Options" registry.

Value	Description	Reference
TBD	VTN Option	this document

7. Security Considerations

TBD

8. Contributors

Zhibo Hu
Email: huzhibo@huawei.com

Lei Bao
Email: baolei7@huawei.com

9. Acknowledgements

The authors would like to thank Juhua Xu and James Guichard for their review and valuable comments.

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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

10.2. Informative References

[I-D.dong-teas-enhanced-vpn-vtn-scalability]
Dong, J., Li, Z., and F. Qin, "Virtual Transport Network (VTN) Scalability Considerations for Enhanced VPN", draft-dong-teas-enhanced-vpn-vtn-scalability-00 (work in progress), February 2020.

[I-D.ietf-teas-enhanced-vpn]
Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A Framework for Enhanced Virtual Private Networks (VPN+) Service", draft-ietf-teas-enhanced-vpn-06 (work in progress), July 2020.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

[TS23501] "3GPP TS23.501", 2016, <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144>.

Authors' Addresses

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Road
Beijing 100095
China

Email: jie.dong@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Campus, No. 156 Beiqing Road
Beijing 100095
China

Email: lizhenbin@huawei.com

Chongfeng Xie
China Telecom
China Telecom Beijing Information Science & Technology, Beiqijia
Beijing 102209
China

Email: xiechf@chinatelecom.cn

Chenhao Ma
China Telecom
China Telecom Beijing Information Science & Technology, Beiqijia
Beijing 102209
China

Email: machh@chinatelecom.cn

Network Working Group T. Herbert
Internet-Draft Intel
Intended status: Experimental October 29, 2020
Expires: May 2, 2021

Attribution Option for Extension Header Insertion
draft-herbert-6man-eh-attrib-03

Abstract

This document defines an "Attribution Option" that provides attribution for IPv6 extension headers, Hop-by-Hop options, or Destination options that are inserted by intermediate nodes in the delivery path of a packet. The purpose of this option is twofold: first it identifies the extension headers or options that have been inserted, secondly it attributes the inserted extension headers or options to the node responsible for inserting them.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Extension header insertion has been proposed as a mechanism to annotate packets for transit across controlled, or limited domains ([I-D.voyer-6man-extension-header-insertion] and [I-D.ietf-ippm-ioam-ipv6-options]). These annotations are in the form of inserted Hop-by-Hop or Destination options, or other inserted extension headers such Segment Routing Header. Presumably, before a packet egresses a controlled domain, any inserted extension headers or options should be removed.

Extension header or options insertion, removal, and other non-standard modifications at intermediate nodes are currently prohibited by [RFC8200], and [I-D.smith-6man-in-flight-eh-insertion-harmful] provides the rationale for why extension header insertion is harmful and thus prohibited. This document addresses the main problem of extension header insertion which is the loss of attribution to the source of packet contents. An "Attribution Option", either as a Hop-by-Hop or Destination option, is defined to provide proper attribution.

The Attribution Option provides two salient benefits:

* The Attribution Option unambiguously identifies what extension headers and Destination or Hop-by-Hop options were inserted by intermediate nodes.

* The Attribution Option includes an identification of the intermediate node that inserted extension headers or options into a packet.

1.1. Motivation for extension header insertion

IP-in-IP encapsulation has been proposed as an alternative to extension header insertion. While encapsulation may be functionally equivalent to header insertion, there are merits to header insertion:

* Extension header insertion can result in fewer bytes of overhead than encapsulation.

* The proper destination address to set in the encapsulating IP header may be unknown. For instance, a node might insert an extension header into an existing packet with the intent that the packet is routed based on the original destination to some arbitrary egress node of the domain and that node removes the inserted headers.

* Packets for a flow may require consistent routing whether or not extension headers are inserted. In particular, to route flows consistently in Equal Cost MultiPath (ECMP), the hash computed for ECMP should be the same for all packets of the flow. Unlike IP encapsulation, extension header insertion shouldn't affect the fields used in ECMP hash calculation (the source address, destination address, flow label, and transport layer ports), so the ECMP hash calculation consistently derives the same value for all packets of a flow with or without inserted extension headers or options.

1.2. Problems with extension header and options insertion

Insertion or removal of extension headers, as well as Destination or Hop-by-Hop options, is currently prohibited by [RFC8200]:

> 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 rationale for this prohibition is articulated in [I-D.smith-6man-in-flight-eh-insertion-harmful]. A summary of cited problems with extension header and options insertion are:

* Extension header and options insertion break the attribution model of IP in that the contents of a packet are no longer attributable to the node identified by the source address of a packet (exceptions include data that a source sets in a packet that is explicitly specified to be modifiable).

* Extension header and options insertion break PMTU discovery since they increase the size of packets in flight.

* Extension header and options insertion break ICMP since inserted extension headers may themselves cause ICMP errors that are sent to the source address. If the source node receives such an ICMP error it cannot take any action to resolve the error since it's not the source of the data that caused the error.

* Extension header and options insertion may create a communications black hole if the data inserted by one node causes a packet to be dropped by a later downstream node. When this happens the source does not know the identity of the node that inserted the data and won't know which node dropped the packet unless an ICMP error is received. In any case, the sending host cannot address the issue, hence persistent, systematic packet loss is possible. Such a scenario may be difficult to troubleshoot in an even moderately large network.

* Use of extension header insertion is generally assumed to be confined to a controlled domain where the domain is a walled garden such that inserted extension headers are always removed before packets would exit a domain. It is conceivable that configuration or implementation errors may allow packets with inserted extension headers to leak out of the controlled domain.

* Extension header and options insertion break the IP Authentication Header (AH) [RFC4302]. If a receiving node attempts to verify an authentication header that covers data inserted by intermediate nodes, then the packet authentication will fail and the packet will be dropped.

This proposal primarily addresses the attribution of packet contents problem. A solution to the attribution problem addresses or at least can mitigate the other problems with extension header insertion.

1.3. Inserting Hop-by-Hop options

Hop-by-Hop options MAY be inserted by intermediate nodes in the delivery path of a packet with the use of the Attribution Option. Hop-by-hop options that have been inserted into a packet, as indicated by the Attribution Option, MAY be removed by intermediate nodes in the delivery path of a packet.

For inserting Hop-by-Hop options into a packet there are two possibilities: 1) a Hop-by-Hop Options extension header already exists in the packet, 2) no Hop-by-Hop Options extension header exist in the packet so a Hop-by-Hop extension header is inserted into the packet which contains the options being inserted.

Note that per [RFC8200] there can only be one Hop-by-Hop Options extension header in a packet, and if present it must be the first extension header after the IPv6 header. If Hop-by-Hop Options are to be inserted into a packet with an existing Hop-by-Hop Options extension header, the options MUST be inserted into the options list for the existing extension header.

1.4. Inserting Destination options

Destination options MAY be inserted into Destination Options before a routing header. Intermediate destination nodes specified in the routing header MAY insert options into Destination Options before the routing header. Other intermediate nodes in the delivery path, specifically those that are not intermediate destinations in the routing header, SHOULD NOT insert Destination options into the Destination Options before the routing header.

Destination options SHOULD NOT be inserted into or removed from Destination options after the routing header or inserted or removed from a packet that does not contain a routing header. The rationale is that intermediate nodes are not supposed to process these Destination options.

An exception to the above rules is that when an extension header is being inserted, the extension header must be preceded by a Destination Options Extension Header that contains the Attribution Option (as described below). In this case, the insertion of a Destination option, precisely only the Attribution Option, is permissible by an intermediate node in the delivery path of a packet. Accordingly, when the inserted extension header is being removed by an intermediate node, the Attribution Option describing it is also removed by the intermediate node.

When inserting Destination options, if an appropriate Destination Options extension header does not exist in the packet then a new Destination Options extension header containing the inserted options is inserted in the packet. The recommended ordering of extension headers in [RFC8200] SHOULD be maintained.

1.5. Inserting extension headers

When an extension header, not Hop-by-Hop or Destination Options, is inserted into a packet it is immediately preceded by a Destination Options extension header that includes an Attribution Option which describes the inserted extension header. If the extension header is being inserted immediately after an existing Destination Options extension header then the Attribution Option is inserted into the existing Destination Options extension header. If there is no preceding Destination Options extension header then one is created into which the Attribution Options is set.

1.6. Scope

This document describes a mechanism for providing attribution in extension header insertion and insertion of Hop-by-Hop and Destination Options. With the exception of inserting Hop-by-Hop Options and Destination Options, requirements and semantics for inserting specific types of extension headers are out of scope. Similarly, security aspects, including potential leakage of inserted headers outside of a controlled domain, are not in scope.

1.7. Requirements Language

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

2. Attribution Option

2.1. Format

The format of the Hop-by-Hop or Destination Attribution Option is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |  Option Type  | Opt Data Len  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|  Num_opts   |                                               |
+-+-+-+-+-+-+-+-+                                               +
|                                                               |
~                        Identification                         ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields are:

* Option Type: value is TBA. The first three bits of the option type should be 000 to indicate that the option is to be skipped over when processed as an unknown option and that the option data is unmodifiable.

* Opt Data Len: data length for the option. The minimal data length is one. If the data length equals twenty then the Identification is an IPv6 address (see section 2.1.2).

* E: For Destination Options this indicates that the extension header following the Destination Options extension header has been inserted. When the option is in Hop-by-Hop Options, this bit MUST be zero when transmitting and ignored on receive.

* Num_opts: If this value is less than 127 then it indicates the number of non-padding options following the Attribution Option that are attributed as being inserted. If the value is 127 then this indicates that the extension header was inserted and all following options are attributed as being inserted. Note that the maximum number of inserted options attributed by one Attribution Option is 126.

* Identification: indicates the source node responsible for the inserted extension headers. This can either be the IPv6 address of the responsible node or a local identifier value that is interpreted by the local network domain (see examples below). Note this field is variable length.

If options are being inserted into an existing Destination Options or Hop-by-Hop Options extension header then the Attribution Option is inserted as the first option in the header, followed by any inserted options, and then followed by any pre-existing options. The total length of the Attribution Option and any inserted options MUST be 8n; this ensures that any pre-existing options following those being inserted retain their original alignment. After the last inserted option, the minimum amount of padding is added to make the total length of inserted data 8n. Pre-existing options, including padding, MUST NOT be modified other than moving them to follow the inserted options.

If a Destination or Hop-by-Hop Options extension header is being inserted in a packet then the Attribution Option is set as the first option in the header followed by any inserted options. Minimal padding MUST added make the length of the extension header 8n.

2.1.1. Attribution Option with short identifier

Below is the short format of the Attribution Option.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |     Type      |       4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|  Num opts   |                  Local_ID                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Local_ID is interpreted locally. For instance, it may be used as an index to a table to map a value to an IPv6 address.

2.1.2. Attribution Option with IPv6 address identifier

Below is the format of the Attribution Option that contains an IPv6 address for attribution of the inserted extension headers or options.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                    |     Type      |      20       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |E|  Num opts   |               Local_ID                        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                                                               +
    |                                                               |
    +                          IPv6 address                         +
    |                                                               |
    +                                                               +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Local_ID contains supplemental identification that is interpreted by the local network. This MAY be the AS of network corresponding to the node identified by the IPv6 address.

2.2. Model

The Attribution Option indicates both inserted Hop-by-Hop or Destination options and inserted extension headers.

Multiple extension header or options insertions may occur during the lifetime of a packet. Insertions are treated as a stack. Hop-by-Hop and Destination options MUST be inserted in an extension header before any pre-existing options including those previously inserted. Similarly, if an extension header is being inserted and a corresponding attribution option is being added to a Destination Option extension header then the inserted extension header immediately follows the Destination Options extension header and precedes any previously inserted extension headers with an Attribution Option in the same Destination Options extension header.

Inserted extension headers and inserted Hop-by-Hop and Destination options MUST be removed in the reverse order of insertion (i.e. inserted data are "popped" to remove them). When an Attribution Option is removed from a packet, which is the first option in the extension header, the option, any corresponding inserted options, and any inserted trailing padding after the last options are removed. In the case of a Destination Options or Hop-by-Hop Options extension header that was inserted, the inserted extension header is removed when the last Attribution Option in the extension header is removed (Num_opts in the option is equal to 127).

The logical structure of an IPv6 packet with inserted extension headers and options, and the relationship between Attribution Options and inserted extension headers and options, is demonstrated below. In this example, a Hop-by-Hop Options extension header was inserted that indicates inserted Hop-by-Hop options. There are two Attribution Options inserted into an existing Destination Options header: the first one (#1) indicates an inserted extension header and no options, the second (#2) indicates an inserted extension header and also inserted Destination options.

```
+-+-+-+-+-+-+-+-+-+
|  IPv6 header    |
+-+-+-+-+-+-+-+-+-+
|  Hop-by-Hop EH  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Attribution Opt |
    +-+-+-+-+-+-+-+-+-+-+-+
    |  Inserted options |
+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  DestOpt EH     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Attribution Opt |---------+    #2 Attribution Option
    +-+-+-+-+-+-+-+-+-+-+-+         |
    |  Inserted options |         |
    +-+-+-+-+-+-+-+-+-+-+-+         |
    |   Attribution Opt |----+    |    #1 Attribution Option
    +-+-+-+-+-+-+-+-+-+-+-+    |    |
    |  Original options |    |    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+    |    |
|  Inserted EH    |<---------+----+
+-+-+-+-+-+-+-+-+-+         |
|  Inserted EH    |<---------+
+--+-+-+-+-+-+-+-+
|  Original EHs   |
+-+-+-+-+-+-+-+-+-+
```

3. Operation

This section describes operations for extension header and options insertion and removal at intermediate nodes.

3.1. Insertion

An extension header or Hop-by-Hop or Destination options MAY be inserted into a packet. The packet's size will increase, and if options are inserted into Destination or Hop-by-Hop Options then the size of those extension headers will increase.

3.1.1. Insertion procedure

Hop-by-Hop and Destination options, including the Attribution Option, are inserted into a packet with the following procedures.

Procedures:

* If an appropriate Hop-by-Hop or Destination Options extension header does not exist in the packet:

  1) Insert a Hop-by-Hop or Destination Options extension header into the packet at the appropriate offset. The extension header contains the Attribution Option, followed by any Hop-by-Hop or Destination options being inserted. Num_opts is set to 127 to indicate that the extension header was inserted. The E bit is set if another extension header is also being inserted (applicable to Destination Options). Add padding to make the length of the extension header be a multiple of eight bytes per [RFC8200].

  2) If no other extension header is being inserted then the nexthdr of the inserted Destination or Hop-by-Hop Options extension header is set to value of the nexthdr in the preceding IPv6 header or extension header.

  3) Else, if an extension header is being inserted then the nexthdr of the inserted Destination Options extension header is set to protocol number of the inserted extension header. The nexthdr of the inserted extension header is set to value of the original nexthdr in the IPv6 header or extension header that precedes the Destination Option being inserted.

  4) The nexthdr of the IPv6 header or extension header that precedes the inserted Destination of Hop-by-Hop Options is set to the protocol number for the inserted header (either 0 for Hop-by-Hop Options or 60 for Destination Options).

* Else, if an appropriate Hop-by-Hop or Destination Options extension header is already present then insert new options into the existing header:

  1) Make first option to be the Attribution Option. Num_opts is set to the number of non-padding options being inserted not including the Attribution Option. The E bit is set if an extension header is being inserted (applicable to Destination Options only).

2) Following the Attribution Option, set any other options being inserted. Include padding before the options as necessary to enforce any alignment requirements.

3) Following the last inserted option, add the minimal amount of padding such that the alignment of the first byte after the last inserted byte is 8n+2 from the start of the Hop-by-Hop or Destination extension header. This is necessary to preserve alignment requirements of existing options. The amount of padding needed is:

   7 - ((offset_last_inserted_byte - 3) % 8)

4) Following the last inserted option and inserted padding, copy the original options from the packet.

5) Set length of the Hop-by-Hop or Destination Options extension header to reflect the length with the inserted options and any inserted padding.

6) If an extension header is being inserted then the nexthdr of the Destination Options header is set to protocol number of the inserted extension header. The nexthdr for the inserted extension header is set to the original nexthdr value of Destination Options extension header.

3.1.2. Errors during insertion

Errors may occur in the process of inserting extension headers in a packet. Error conditions would include the resultant packet size exceeding MTU, and the size of Hop-by-Hop Options extension header exceeding 1024 bytes (the maximum size of the Hop-by-Hop Options extension header).

If an error occurs during insertion then the node performing insertion MUST take an appropriate behavior per some configuration. The packet MAY be discarded or the unmodified packet MAY be forwarded. An error SHOULD be logged.

3.2. Removal of inserted extension headers and options

The top level inserted extension headers and Hop-by-Hop or Destination options, referred to by the Attribution Option which is the first option in the Hop-by-Hop or Destination options of a packet, MAY be removed by an intermediate node.

3.2.1. Removal procedure

The procedure is:

* If Num_opts equals 127 then the Destination or Hop-by-Hop extension header is to be removed.

  * If the E bit is not set or a Hop-by-Hop extension header is being removed, remove the Destination or Hop-by-Hop Options extension header bytes from the packet and set the nexthdr of the preceding IPv6 header or extension header to the nexthdr of the Destination or Hop-by-Hop Options extension header being removed.

  * Else, if the E bit is set in the Attribution Option of a Destination Options extension header, remove the extension header bytes of the Destination Options extension header and those of the extension header following the Destination Options extension header from the packet. The nexthdr of the preceding IPv6 header or extension header is set to the nexthdr of the of the extension header following the Destination Options extension header.

* Else, if Num_opts is less than 127, then the inserted options must be removed from the existing header:

  1) Locate the last inserted option. This done by the scanning non-padding options after the Attribution Option for the count in Num_opts.

  2) Compute the amount of padding that was inserted. The amount of padding that should have been inserted is:

     7 - ((offset_last_inserted_byte - 2) % 8)

     where offset_last_byte is the offset of the last byte of the last inserted option located in step #1.

  3) Remove the bytes in the packet from first byte of the Destination or Hop-by-Hop Options data (first byte of the Attribution option) through the last byte of inserted padding as computed in step #2.

  4) Set the length of the Hop-by-Hop Options extension header to account for the removed bytes; that is the original extension header length minus the number of removed bytes.

5) If the E bit is set in the Attribution Option being removed from a Destination Options extension header, remove the following extension header from the packet. The nexthdr of the Destination Options extension header is set to the nexthdr of the extension header being removed.

3.2.2. Errors during removal

A node performing extension header removal MUST validate packet contents.

The following attributes MUST be validated before removal:

* If Num_opts is not equal to 127 then number of non-padding options following Attribution Option MUST be greater than or equal to Num_opts.

* Necessary padding after the last inserted Hop-by-Hop option MUST be present. The amount of padding MUST be equal to the expected amount.

* The Num_opts options following the Attribution Option MUST NOT contain another Attribution Option.

* If the E bit is set in the Attribution options of a Destination Options header then the a valid extension header MUST follow the Destination Options header.

If any of the above validations fail, or an error is otherwise encountered in the removal process, then the processing node MUST take action. The packet SHOULD be discarded and error message SHOULD be logged.

3.3. Domain edge filtering

Filtering packets with inserted extension headers or Destination or Hop-by-Hop options is straightforward: a packet contains inserted options if the first option of a Destination Options or Hop-by-Hop Options is the Attribution Option. A packet contains inserted extension headers if it contains an Attribution Option in Destination Options with the E bit set or the packet contains a Destination Options or Hop-by-Hop Options extension header that includes an Attribution Option with Num_opts equal to 127 (in which case the containing Destination Options or Hop-by-Hop extension header was inserted).

3.4. ICMP processing

At described in [I-D.smith-6man-in-flight-eh-insertion-harmful], it is possible for a source node to receive ICMP [RFC4443] errors caused by inserted headers, thus the source node has no recourse to address the error.

This section proposes some ways to apply the Attribution Option to mitigate the ICMP breakage for extension header insertion:

* ICMP errors can be filtered [RFC4890] by nodes in the network before reaching a source node outside of the domain (at the domain edge for instance). The packet headers in the ICMP data should include the Destination Options or Hop-by-Hop Options extension header containing the Attribution Option. The filtering node MAY analyze the error to determine if it was caused by the inserted headers:

  - If the error was caused by inserted extension headers, then the node SHOULD take appropriate actions (minimally it SHOULD log the error). The filtering node SHOULD not forward the ICMP error to the source.

  - If the error was not caused by inserted headers, the filtering node MAY create a new ICMP error with the data packet that would be reflect the packet contents prior to extension header insertion (i.e. attempt set the packet in ICMP to be that which the source would have sent). This is done by removing the inserted extension headers of the packet in the ICMP data, and adjusting the Pointer field in an ICMP error if necessary. The revised ICMP error can then be forwarded to the source.

* If ICMP errors are not filtered and the source node receives an ICMP error for a packet containing inserted extension headers:

  - If the source node is a legacy implementation that does not understand the Attribution Option then it will attempt to process the error under the assumption that it was the source of the packet and the data that caused the error. If the node logs the contents of the ICMP error, which should be common, then external out-of-band analysis can be done by network administrators to troubleshoot the ICMP errors and identify culprit if the error was caused by inserted extension headers.

  - If the source node understands the Attribution Option then it can perform more analysis. The node MAY attempt to ascertain if the error was caused by inserted headers or not, and if not it can then attempt to fix the problem with the assumption the it was responsible for the data in error.

3.5. Processing AH

Extension headers and options MAY be inserted into a packet before an existing AH header. The inserted data is not covered in the ICV computation and if a receiving host attempts to perform the ICV computation over inserted data it is expected that verification will fail and the packet will be dropped.

The simplest way to address this is to remove any inserted headers in the packet before processing the AH extension header. The assumption is that once the inserted data is removed, the packet contents reflect the original contents set by the host so AH verification should succeed.

Host implementations can be modified to process the attribution option. When a packet with inserted headers or options is received by an end host, the AH processing can ignore any inserted Destination or Hop-by-Hop options and any inserted extension headers. This can be done in conjunction with the existing algorithms to ignore option data in the ICV computation for modifiable options. Effectively, the algorithm is simply to remove all the inserted options and extension headers following the procedures in section 3.1.

4. Security Considerations

The Attribution Option does not in itself introduce any new security considerations. The security of containing inserted extension headers within a controlled domain is out of scope for this document.

Section 3.5 describes the processing of the IP Authentication Header in the presence of inserted options or extension headers.

5. IANA Considerations

IANA is requested to assigned the following Destination and Hop-By-Hop option:

Hex Value	Binary value act chg rest	Description	Reference
TBD	00 0 TBD	Attribution Option	This document

6. References

6.1. 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>.

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

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

6.2. Informative References

[I-D.ietf-ippm-ioam-ipv6-options] Bhandari, S., Brockners, F., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Kfir, A., Gafni, B., Lapukhov, P., Spiegel, M., Krishnan, S., Asati, R., and M. Smith, "In-situ OAM IPv6 Options", draft-ietf-ippm-ioam-ipv6-options-03 (work in progress), September 2020.

[I-D.smith-6man-in-flight-eh-insertion-harmful] Smith, M., Kottapalli, N., Bonica, R., Gont, F., and T. Herbert, "In-Flight IPv6 Extension Header Insertion Considered Harmful", draft-smith-6man-in-flight-eh-insertion-harmful-02 (work in progress), May 2020.

[I-D.voyer-6man-extension-header-insertion] Voyer, D., Filsfils, C., Dukes, D., Matsushima, S., Leddy, J., Li, Z., and J. Guichard, "Deployments With Insertion of IPv6 Segment Routing Headers", draft-voyer-6man-extension-header-insertion-09 (work in progress), May 2020.

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

Author's Address

Tom Herbert
Intel
Santa Clara, CA
USA

Email: tom@quantonium.net

IPv6 Maintenance
Internet-Draft
Updates: 4861 (if approved)
Intended status: Standards Track
Expires: March 19, 2021
J. Linkova
Google
September 15, 2020

Gratuitous Neighbor Discovery: Creating Neighbor Cache Entries on First-Hop Routers

draft-ietf-6man-grand-03

Abstract

Neighbor Discovery (RFC4861) is used by IPv6 nodes to determine the link-layer addresses of neighboring nodes as well as to discover and maintain reachability information. This document updates RFC4861 to allow routers to proactively create a Neighbor Cache entry when a new IPv6 address is assigned to a node. It also updates RFC4861 and recommends nodes to send unsolicited Neighbor Advertisements upon assigning a new IPv6 address. The proposed change will minimize the delay and packet loss when a node initiate connections to off-link destination from a new IPv6 address.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Neighbor Discovery state machine defined in [RFC4861] assumes that communications between IPv6 nodes are in most cases bi-directional and if a node A is trying to communicate to its neighbor, node B, the return traffic flows could be expected. So when the node A starts the address resolution process, the target node B would also create an entry for A address in its neighbor cache. That entry will be used for sending the return traffic to A.

In particular, section 7.2.5 of [RFC4861] states: "When a valid Neighbor Advertisement is received (either solicited or unsolicited), the Neighbor Cache is searched for the target's entry. If no entry exists, the advertisement SHOULD be silently discarded. There is no need to create an entry if none exists, since the recipient has apparently not initiated any communication with the target."

While this approach is perfectly suitable for host-to-host on-link communications, it does not work so well when a host sends traffic to off-link destinations. After joining the network and receiving a Router Advertisement the host populates its neighbor cache with the default router IPv6 and link-layer addresses and is able to send traffic to off-link destinations. At the same time the router does not have any cache entries for the host global addresses yet and only starts address resolution upon receiving the first packet of the return traffic flow. While waiting for the resolution to complete routers only keep a very small number of packets in the queue, as recommended in Section 7.2.2 [RFC4861]. All subsequent packets arriving before the resolution process finishes are likely to be dropped. It might cause user-visible packet loss and performance degradation.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

Node: a device that implements IP, [RFC4861].

Host: any node that is not a router, [RFC4861].

ND: Neighbor Discovery, [RFC4861].

SLAAC: IPv6 Stateless Address Autoconfiguration, [RFC4862].

NS: Neighbor Solicitation, [RFC4861].

NA: Neighbor Advertisement, [RFC4861].

RS: Router Solicitation, [RFC4861].

RA: Router Advertisement, [RFC4861].

SLLA: Source link-layer Address, an option in the ND packets containing the link-layer address of the sender of the packet [RFC4861].

TLLA: Target link-layer Address, an option in the ND packets containing the link-layer address of the target [RFC4861].

GUA: Global Unicast Address [RFC4291].

DAD: Duplicate Address Detection, [RFC4862].

Optimistic DAD: a modification of DAD, [RFC4429].

2. Problem Statement

The most typical scenario when the problem may arise is a host joining the network, forming a new address and using that address for accessing the Internet:

1. A host joins the network and receives a Router Advertisement (RA) packet from the first-hop router (either a periodic unsolicited RA or a response to a Router Solicitation sent by the host). The RA contains information the host needs to perform SLAAC and to configure its network stack. The RA is send from the router’s link-local address to link-local destination and may contain the link-layer address of the router. As a result the host can populate its Neighbor Cache with the router’s link-local and link-layer addresses.

2. The host starts opening connections to off-link destinations. A very common use case is a mobile device sending probes to detect the Internet connectivity and/or the presence of a captive portal on the network. To speed up that process many implementations use Optimistic DAD which allows them to send probes before the DAD process is completed. At that moment the device neighbor cache contains all information required to send those probes (such as the default router link-local the link-layer addresses). The router neighbor cache, however, might contain an entry for the device link-local address (if the device has been performing the address resolution for the router link-local address), but there are no entries for the device global addresses.

3. Return traffic is received by the first-hop router. As the router does not have any cache entry for the host global address yet, the router starts the neighbor discovery process by creating an INCOMPLETE cache entry and then sending a Neighbor Solicitation to the Solicited Node Multicast Address. As per Section 7.2.2 of [RFC4861] Routers MUST buffer at least one data packet and MAY buffer more, while resolving the packet destination address. However most router implementations limit the buffer size to a few packets only, so all subsequent packets for the host global address are dropped, until the address resolution process is completed.

4. If the host sends multiple probes in parallel, it would consider all but one of them failed. That leads to user-visible delay in connecting to the network, especially if the host implements some form of backoff mechanism and does not retransmit the probes as soon as possible.

This scenario illustrates the problem occurring when the device connects to the network for the first time or after an inactivity period long enough for the device address to be removed from the router's neighbor cache. However, the same sequence of events happen when the host starts using a new global address previously unseen by the router, such as a new privacy address [RFC4941] or if the router's Neighbor Cache has been flushed.

While in dual-stack networks this problem might be hidden by Happy Eyeballs [RFC8305] it manifests quite clearly in IPv6-only environments, especially wireless ones, leading to poor user experience and contributing to a negative perception of IPv6-only solutions as unstable and non-deployable.

3. Solution Requirements

It would be highly desirable to improve the Neighbor Discovery mechanics so routers have a usable cache entry for a host address by the time the router receives the first packet for that address. In particular:

- o If the router does not have a Neighbor Cache entry for the address, a STALE entry needs to be created.

- o The solution needs to work for Optimistic addresses as well. Devices implementing the Optimistic DAD usually attempt to minimize the delay in connecting to the network and therefore are more likely to be affected by the problem described in this document.

- o In case of duplicate addresses present in the network, the proposed solution MUST NOT override the existing entry.

- o In topologies with multiple first-hop routers the cache needs to be updated on all of them, as traffic might be asymmetric: outgoing flows leaving the network via one router while the return traffic enters the segment via another one.

In addition the solution MUST NOT exacerbate issues described in [RFC6583] and MUST be compatible with the recommendations provided in [RFC6583].

4. Proposed Changes to Neighbor Discovery

The following changes are proposed to minimize the delay in creating new entries in a router neighbor cache

- o A node sends unsolicited NAs upon assigning a new IPv6 address to its interface.

- o A router creates a new cache entry upon receiving an unsolicited NA from a host.

The following sections discuss these changes in more detail.

4.1. Nodes Sending Gratuitous Neighbor Advertisements

The section 7.2.6 of [RFC4861] discusses using unsolicited Neighbor Advertisement to inform node neighbors of the new link-layer address quickly. The same mechanism could be used to notify the node neighbors about the new network-layer address as well: the node can send gratuitous unsolicited Neighbor Advertisements upon assigning a new IPv6 address to its interface.

To minimize the potential disruption in case of duplicate addresses the node should not set the Override flag for a preferred address and must not set the Override flag if the address is in Optimistic [RFC4429] state.

As the main purpose of sending unsolicited NAs upon configuring a new address is to proactively create a Neighbor Cache entry on the first-hop routers, the gratuitous NAs are sent to all-routers multicast address (ff02::2). Limiting the recipients to routers only would help reduce the multicast noise level. If the link-layer devices are performing MLD snooping [RFC4541] then those unsolicited NAs will be only sent to onlink routers instead of being flooded to all nodes.

It should be noted that the proposed mechanism does not cause any significant increase in the multicast traffic. The additional multicast unsolicited NA would proactively create a STALE cache entry on routers as discussed below. When the router receives the return traffic flows it does not need to send multicast NSes to the solicited node multicast address but would be sending unicast NSes instead. Therefore total amount of multicast traffic should not increase.

4.2. Routers Creating Cache Entries Upon Receiving Unsolicited Neighbor Advertisements

The section 7.2.5 of [RFC4861] states: "When a valid Neighbor Advertisement is received (either solicited or unsolicited), the Neighbor Cache is searched for the target's entry. If no entry exists, the advertisement SHOULD be silently discarded. There is no need to create an entry if none exists, since the recipient has apparently not initiated any communication with the target".

The reasoning behind dropping unsolicited Neighbor Advertisements ("the recipient has apparently not initiated any communication with the target") is valid for onlink host-to-host communication but, as discussed above, it does not really apply for the scenario when the host is announcing its address to routers. Therefore it would be beneficial to allow routers creating new entries upon receiving an unsolicited Neighbor Advertisement.

This document updates [RFC4861] so that routers create a new Neighbor Cache entry upon receiving an unsolicited Neighbor Advertisement. The proposed changes do not modify routers behaviour specified in [RFC4861] for the scenario when the corresponding Neighbor Cache entry already exists.

5. Avoiding Disruption

If nodes following the recommendations in this document are using the DAD mechanism defined in [RFC4862], they would send unsolicited NA as soon as the address changes the state from tentative to preferred (after its uniqueness has been verified). However nodes willing to minimize network stack configuration delays might be using optimistic addresses, which means there is a possibility of the address not being unique on the link. The section 2.2 of [RFC4429] discusses measures to ensure that ND packets from the optimistic address do not override any existing neighbor cache entries as it would cause traffic interruption of the rightful address owner in case of address conflict. As nodes willing to speed up their network stack configuration are most likely to be affected by the problem outlined in this document it seems reasonable for such hosts to advertise their optimistic addresses by sending unsolicited NAs. The main question to consider is the potential risk of overriding the cache entry for the rightful address owner if the optimistic address happens to be duplicated.

The following sections are discussing the address collision scenario when a node sends an unsolicited NA for an address in the Optimistic state, while another node has the same address assigned already.

5.1. Neighbor Cache Entry Exists in Any State Other That INCOMPLETE

If the router Neighbor Cache entry for the target address already exists in any state other than INCOMPLETE, then as per section 7.2.5 of [RFC4861] an unsolicited NA with the Override flag cleared would change the entry state from REACHABLE to STALE but would not update the entry in any other way. Therefore even if the host sends an unsolicited NA from the its Optimistic address the router cache entry would not be updated with the new Link-Layer address and no impact to the traffic for the rightful address owner is expected.

5.2. Neighbor Cache Entry is in INCOMPLETE state

Another corner case is the INCOMPLETE cache entry for the address. If the host sends an unsolicited NA from the Optimistic address it would update the entry with the host link-layer address and set the entry to the STALE state. As the INCOMPLETE entry means that the router has started the ND process for the address and the multicast NS has been sent, the rightful owner is expected to reply with solicited NA with the Override flag set. Upon receiving a solicited NA with the Override flag the cache entry will be updated with the TLLA supplied and (as the NA has the Solicited flag set), the entry state will be set to REACHABLE. It would recover the cache entry and set the link-layer address to the one of the rightful owner. The only potential impact would be for packets arriving to the router after the unsolicited NA from the host but before the rightful owner responded with the solicited NA. Those packets would be sent to the host with the optimistic address instead of its rightful owner. However those packets would have been dropped anyway as until the solicited NA is received the router can not send the traffic.

5.3. Neighbor Cache Entry Does Not Exist

There are two distinct scenarios which can lead to the situation when the router does not have a NC entry for the IPv6 address:

1. The rightful owner of the address has not been using it for communication.

2. The rightful owner just started sending packets from that address but the router has not received any return traffic yet.

The impact on the rightful owner’s traffic flows would be different in those cases.

5.3.1. The Rightful Owner Is Not Sending Packets From The Address

In this scenario the following events are expected to happen:

1. The host configures the address and sets its state to Optimistic.

2. The host sends an unsolicited NA with the Override flag set to zero and starts sending traffic from the Optimistic address.

3. The router creates a STALE entry for the address and the host link-layer address.

4. The host starts DAD and detects the address duplication.

5. The router receives the return traffic for the duplicated address. As the NC entry is STALE it sends traffic using that entry, changes it to DELAY and wait up to DELAY_FIRST_PROBE_TIME ([RFC4861]) seconds.

6. The router changes the NC entry state to PROBE and sends up to MAX_UNICAST_SOLICIT ([RFC4861]) unicast NSes separated by RetransTimer milliseconds ([RFC4861]) to the host link-layer address.

7. As the host has detected the address conflict already it does not respond to the unicast NSes.

8. The router sends a multicast NS to the solicited node multicast address, the rightful owner responds and the router NC entry is updated with the rightful owner link-local address.

The rightful owner is not experiencing any disruption as it does not send/receive any traffic. If after step 7 the router keeps receiving any return traffic for communication initiated at step 2, those packets would be forwarded to the rightful owner. However the same behaviour would be observed if changes proposed in this document are implemented: if the host starts sending packets from its Optimistic address but then changed the address state to Duplicated, almost all return traffic would be forwarded to the rightful owner of the said address. Therefore it’s safe to conclude that the proposed changes do not cause any disruption for the rightful owner.

5.3.2. The Rightful Owner Has Started Sending Packets From The Address

In this scenario the following events are happening:

1. The rightful owner starts sending traffic from the address (e.g. the address has just been configured or has not been recently used).

2. The host configures the address and sets its state to Optimistic.

3. The host sends an unsolicited NA with the Override flag set to zero and starts sending traffic from the Optimistic address.

4. The router creates a STALE entry for the address and the host link-layer address.

5. The host starts DAD and detects the address duplication.

6. The router receives the return traffic flows for both the rightful owner of the duplicated address and the new host. As the NC entry is STALE it sends traffic using that entry, changes it to DELAY and wait up to DELAY_FIRST_PROBE_TIME ([RFC4861]) seconds.

7. The router changes the NC entry state to PROBE and sends up to MAX_UNICAST_SOLICIT ([RFC4861]) unicast NSes separated by RetransTimer milliseconds ([RFC4861]) to the host link-layer address.

8. As the host has detected the address conflict already it does not respond to the unicast NSes.

9. The router sends a multicast NS to the solicited node multicast address, the rightful owner responds and the router NC entry is updated with the rightful owner link-local address.

As a result the traffic for the address rightful owner would be sent to the host with the duplicated address instead. The duration of the disruption can be estimated as DELAY_FIRST_PROBE_TIME*1000 + (MAX_UNICAST_SOLICIT - 1)*RetransTimer milliseconds. As per the constants defined in Section 10 of [RFC4861] this interval is equal to 5*1000 + (3 - 1)*1000 = 7000ms or 7 seconds.

However it should be noted that the probability of such scenario is rather low as it would require the following things to happen almost simultaneously (within tens of milliseconds):

- One host starts using a new IPv6 address and sending traffic.

- Another host configures the same IPv6 address in Optimistic mode before the router receives the return traffic for the first host.

6. Modifications to RFC-Mandated Behavior

All normative text in this memo is contained in this section.

6.1. Modification to RFC4861 Neighbor Discovery for IP version 6 (IPv6)

6.1.1. Modification to the section 7.2.5

This document proposes the following changes to the section 7.2.5 of [RFC4861]:

OLD TEXT:

When a valid Neighbor Advertisement is received (either solicited or unsolicited), the Neighbor Cache is searched for the target's entry. If no entry exists, the advertisement SHOULD be silently discarded. There is no need to create an entry if none exists, since the recipient has apparently not initiated any communication with the target.

NEW TEXT:

When a valid Neighbor Advertisement is received (either solicited or unsolicited), the Neighbor Cache is searched for the target's entry. If no entry exists, hosts SHOULD silently discard the advertisement. There is no need to create an entry if none exists, since the recipient has apparently not initiated any communication with the target. Routers SHOULD create a new entry for the target address with the link-layer address set to the Target link-layer address option (if supplied). The entry its reachability state MUST also be set to STALE. If the received Neighbor Advertisement does not contain the Target link-layer address option the advertisement SHOULD be silently discarded.

6.1.2. Modification to the section 7.2.6

This document proposes the following changes to the section 7.2.6 of [RFC4861]:

OLD TEXT:

Also, a node belonging to an anycast address MAY multicast unsolicited Neighbor Advertisements for the anycast address when the node's link-layer address changes.

NEW TEXT:

Also, a node belonging to an anycast address MAY multicast unsolicited Neighbor Advertisements for the anycast address when the node's link-layer address changes.

A node may also wish to notify its first-hop routers when it configures a new global IPv6 address so the routers can proactively populate their neighbor caches with the corresponding entries. In such cases a node SHOULD send up to MAX_NEIGHBOR_ADVERTISEMENT Neighbor Advertisement messages. If the address is preferred then the Override flag SHOULD NOT be set. If the address is in the Optimistic state then the Override flag MUST NOT be set. The destination address SHOULD be set to the all-routers multicast address. These advertisements MUST be separated by at least

RetransTimer seconds. The first advertisement SHOULD be sent as soon as one of the following events happens:

- o if Optimistic DAD [RFC4429] is used: a new Optimistic address is assigned to the node interface.
- o if Optimistic DAD is not used: an address changes the state from tentative to preferred.

7. Solution Limitations

The solution described in this document provides some improvement for a node configuring a new IPv6 address and start sending traffic from it. However that approach does not completely eliminate the scenario when a router receives some transit traffic for an address without the corresponding Neighbor Cache entry. For example:

- o If the host starts using an already configured IPv6 address after a long period of inactivity, the router might not have the NC entry for that address anymore, as old/expired entries are deleted.
- o Clearing the router Neighbor Cache would trigger the packet loss for all actively used addresses removed from the cache.

8. Solutions Considered but Discarded

There are other possible approaches to address the problem, for example:

- o Just do nothing.
- o Migrating from the "reactive" Neighbor Discovery ([RFC4861]) to the registration-based mechanisms ([RFC8505]).
- o Creating new entries in routers Neighbor Cache by gleaning from Neighbor Discovery DAD messages.
- o Initiates bidirectional communication from the host to the router using the host GUA.
- o Making the probing logic on hosts more robust.
- o Increasing the buffer size on routers.

- o Transit dataplane traffic from an unknown address (an address w/o the corresponding neighbor cache entry) triggers an address resolution process on the router.

It should be noted that some of those options are already implemented by some vendors. The following sections discuss those approaches and the reasons they were discarded.

8.1. Do Nothing

One of the possible approaches might be to declare that everything is working as intended and let the upper-layer protocols to deal with packet loss. The obvious drawbacks include:

- o Unhappy users.

- o Many support tickets.

- o More resistance to deploy IPv6 and IPv6-Only networks.

8.2. Change to the Registration-Based Neighbor Discovery

The most radical approach would be to move away from the reactive ND as defined in [RFC4861] and expand the registration-based ND ([RFC6775], [RFC8505]) used in Low-Power Wireless Personal Area Networks (6LoWPANs) to the rest of IPv6 deployments. This option requires some investigation and discussion. However the implementation complexity and unclear adoption timeline makes this approach less preferable than one proposed in this document.

8.3. Host Sending NS to the Router Address from Its GUA

The host could force creating a STALE entry for its GUA in the router ND cache by sending the following Neighbor Solicitation message:

- o The NS source address is the host GUA.

- o The destination address is the default router IPv6 address.

- o The Source Link-Layer Address option contains the host link-layer address.

- o The target address is the host default router address (the default router address the host received in the RA).

The main disadvantages of this approach are:

o Would not work for Optimistic addresses as section 2.2 of [RFC4429] explicitly prohibits sending Neighbor Solicitations from an Optimistic Address.

o If first-hop redundancy is deployed in the network, the NS would reach the active router only, so all backup routers (or all active routers except one) would not get their neighbor cache updated.

o Some wireless devices are known to alter ND packets and perform various non-obvious forms of ND proxy actions. In some cases, unsolicited NAs might not even reach the routers.

8.4. Host Sending Router Solicitation from its GUA

The host could send a router solicitation message to 'all routers' multicast address, using its GUA as a source. If the host link-layer address is included in the Source Link-Layer Address option, the router would create a STALE entry for the host GUA as per the section 6.2.6 of [RFC4861]. However, this approach can not be used if the GUA is in optimistic state: section 2.2 of [RFC4429] explicitly prohibits using an Optimistic Address as the source address of a Router Solicitation with a SLLAO as it might disrupt the rightful owner of the address in the case of a collision. So for the optimistic addresses the host can send an RS without SLLAO included. In that case the router may respond with either a multicast or a unicast RA (only the latter would create a cache entry).

This approach has the following drawbacks:

o If the address is in the Optimistic state the RS can not contain SLLAO. As a result the router would only create a cache entry if solicited RAs are sent as unicast. Routers sending solicited RAs as multicast would not create a new cache entry as they do not need to send a unicast packet back to the host.

o There might be a random delay between receiving an RS and sending a unicast RA back (and creating a cache entry) which might undermine the idea of creating the cache entry proactively.

o Some wireless devices are known to intercept ND packets and perform various non-obvious forms of ND proxy actions. In some cases the RS might not even reach the routers.

8.5. Routers Populating Their Caches by Gleaning From Neighbor Discovery Packets

Routers may be able to learn about new addresses by gleaning from the DAD Neighbor Solicitation messages. The router could listen to all solicited node multicast address groups and upon receiving a Neighbor Solicitation from the unspecified address search its Neighbor Cache for the solicitation's Target Address. If no entry exists, the router may create an entry, set its reachability state to 'INCOMPLETE' and start the address resolution for that entry.

The same solution was proposed in [I-D.halpern-6man-nd-pre-resolve-addr]. Some routing vendors support such optimization already. However, this approach has a number of drawbacks and therefore should not be used as the only solution:

- o Routers need to receive all multicast Neighbor Discovery packets which might negatively impact the routers CPU.

- o If the router starts the address resolution as soon as it receives the DAD Neighbor Solicitation the host might be still performing DAD and the target address might be tentative. In that case, the host SHOULD silently ignore the received Neighbor Solicitation from the router as per the Section 5.4.3 of [RFC4862]. As a result the router might not be able to complete the address resolution before the return traffic arrives.

8.6. Initiating Hosts-to-Routers Communication

The host may force the router to start address resolution by sending a data packet such as ping or traceroute to its default router link-local address, using the GUA as a source address. As the RTT to the default router is lower than RTT to any off-link destinations it's quite likely that the router would start the neighbor discovery process for the host GUA before the first packet of the returning traffic arrives.

This approach has the following drawbacks:

- o Data packets to the router link-local address could be blocked by security policy or control plane protection mechanism.

- o It introduces an additional overhead for routers control plane (in addition to processing ND packets, the data packet needs to be processed as well).

o Unless the data packet is sent to 'all routers' ff02::2 multicast address, if the network provides a first-hop redundancy then only the active router would create a new cache entry.

8.7. Making the Probing Logic on Hosts More Robust

Theoretically the probing logic on hosts might be modified to deal better with initial packet loss. For example, only one probe can be sent or probes retransmit intervals can be reduced. However, this approach has a number of drawbacks:

o It would require updating all possible applications performing probing, while the proposed solution is implemented on operating systems level.

o Some implementations need to send multiple probes. Examples are including but not limited to:

  * Sending AAAA and A records DNS probes in parallel.

  * Detecting captive portals often require sending multiple packets.

o While it would increase the probability of the probing to complete successfully, there are multiple cases when packet loss would still occur:

  * The probe response consists of multiple packets, so all but the first one are dropped.

  * There are multiple applications on the same host sending traffic and return packets arrive simultaneously.

  * There are multiple first-hop routers in the network. The first probe packet creates the NC entry on one of them. The subsequent return traffic flows might cross other routers and still experince the issue.

8.8. Increasing the Buffer Size on Routers

Increasing the buffer size and buffering more packets would exacerbate issues described in [RFC6583] and make the router more vulnerable to ND-based denial of service attacks.

8.9. Transit Dataplane Traffic From a New Address Triggering Address Resolution

When a router receives a transit packet, it might check the presence of the neighbor cache entry for the packet source address and if the entry does not, exist start address resolution process. This approach does ensure that a Neighbor Cache entry is proactively created every time a new, previously unseen GUA is used for sending offlink traffic. However this approach has a number of limitations, in particular:

- o If traffic flows are asymmetrical the return traffic might not transit the same router as the original traffic which triggered the address resolution. So the neighbor cache entry is created on the "wrong" router, not the one which actually needs the neighbor cache entry for the host address.

- o The functionality needs to be limited to explicitly configured networks/interfaces, as the router needs to distinguish between onlink addresses (ones the router needs to have Neighbor Cache entries for) and the rest of the address space.

- o Implementing such functionality is much more complicated than all other solutions as it would involve complex data-control planes interaction.

9. IANA Considerations

This memo asks the IANA for no new parameters.

10. Security Considerations

One of the potential attack vectors to consider is a cache spoofing when the attacker might try to install a cache entry for the victim's IPv6 address and the attacker's Link-Layer address. However it should be noted that this document does not propose any changes for the scenario when the ND cache for the given IPv6 address already exists. Therefore it is not possible for the attacker to override any existing cache entry.

A malicious host could attempt to exhaust the neighbor cache on the router by creating a large number of STALE entries. However this attack vector is not new and this document does not increase the risk of such an attack: the attacker could do it, for example, by sending a NS or RS packet with SLLAO included. All recommendations from [RFC6583] still apply.

Announcing a new address to all-routers multicast address may inform an on-link attacker about IPv6 addresses assigned to the host. However hiding information about the specific IPv6 address should not be considered a security measure as such information is usually disclosed via DAD to all nodes anyway. Network administrators can also mitigate this issue by enabling MLD snooping on the link-layer devices to prevent IPv6 link-local multicast packets being flooded to all onlink nodes. If peer-to-peer onlink communications are not desirable for the given network segment they should be prevented by proper layer2 security mechanisms. Therefore the risk of allowing hosts to send unsolicited Neighbor Advertisements to all-routers multicast address is low.

It should be noted that the proposed mechanism allows hosts to proactively inform their routers about global IPv6 addresses existing on-link. Routers could use that information to distinguish between used and unused addresses to mitigate ND cache exhaustion DoS attacks described in Section 4.3.2 [RFC3756] and [RFC6583].

11. Acknowledgements

Thanks to the following people (in alphabetical order) for their comments, review and feedback: Mikael Abrahamsson, Stewart Bryant, Lorenzo Colitti, Owen DeLong, Igor Gashinsky, Fernando Gont, Tatuya Jinmei, Erik Kline, Warren Kumari, Barry Leiba, Jordi Palet Martinez, Erik Nordmark, Michael Richardson, Mark Smith, Dave Thaler, Pascal Thubert, Loganaden Velvindron, Eric Vyncke.

12. References

12.1. 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>.

[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>.

[RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, DOI 10.17487/RFC4429, April 2006, <https://www.rfc-editor.org/info/rfc4429>.

[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, <https://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, <https://www.rfc-editor.org/info/rfc4862>.

[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, <https://www.rfc-editor.org/info/rfc6775>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/info/rfc8305>.

[RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <https://www.rfc-editor.org/info/rfc8505>.

12.2. Informative References

[I-D.halpern-6man-nd-pre-resolve-addr] Chen, I. and J. Halpern, "Triggering ND Address Resolution on Receiving DAD-NS", draft-halpern-6man-nd-pre-resolve-addr-00 (work in progress), January 2014.

[RFC3756] Nikander, P., Ed., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, DOI 10.17487/RFC3756, May 2004, <https://www.rfc-editor.org/info/rfc3756>.

[RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <https://www.rfc-editor.org/info/rfc4541>.

[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>.

[RFC6583] Gashinsky, I., Jaeggli, J., and W. Kumari, "Operational Neighbor Discovery Problems", RFC 6583, DOI 10.17487/RFC6583, March 2012, <https://www.rfc-editor.org/info/rfc6583>.

Author's Address

Jen Linkova
Google
1 Darling Island Rd
Pyrmont, NSW 2009
AU

Email: furry@google.com

6MAN Working Group G. Fioccola
Internet-Draft T. Zhou
Intended status: Standards Track Huawei
Expires: April 16, 2021 M. Cociglio
Telecom Italia
F. Qin
China Mobile
R. Pang
China Unicom
October 13, 2020

IPv6 Application of the Alternate Marking Method
draft-ietf-6man-ipv6-alt-mark-02

Abstract

This document describes how the Alternate Marking Method can be used as the passive performance measurement tool in an IPv6 domain and reports implementation considerations. It proposes how to define a new Extension Header Option to encode alternate marking technique and both Hop-by-Hop Options Header and Destination Options Header are considered.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Status of This Memo

Table of Contents

1. Introduction

[RFC8321] and [RFC8889] describe a passive performance measurement method, which can be used to measure packet loss, latency and jitter on live traffic. Since this method is based on marking consecutive batches of packets, the method is often referred as Alternate Marking Method.

The Alternate Marking Method has become mature to be implemented and encoded in the IPv6 protocol and this document defines how it can be used to measure packet loss and delay metrics in IPv6.

The format of the IPv6 addresses is defined in [RFC4291] while [RFC8200] defines the IPv6 Header, including a 20-bit Flow Label and the IPv6 Extension Headers. The Segment Routing Header (SRH) is defined in [RFC8754] to apply Segment Routing over IPv6 dataplane (SRv6).

[I-D.fioccola-v6ops-ipv6-alt-mark] reported a summary on the possible implementation options for the application of the Alternate Marking Method in an IPv6 domain. This document, starting from the outcome of [I-D.fioccola-v6ops-ipv6-alt-mark], introduces a new TLV that can be encoded in the Options Headers (both Hop-by-Hop or Destination) for the purpose of the Alternate Marking Method application in an IPv6 domain. The case of SRH ([RFC8754]) is also discussed, anyway this is valid for all the types of Routing Header (RH).

2. Alternate Marking application to IPv6

The Alternate Marking Method requires a marking field. As mentioned, several alternatives have been analysed in [I-D.fioccola-v6ops-ipv6-alt-mark] such as IPv6 Extension Headers, IPv6 Address and Flow Label.

In consequence to the previous document and to the discussion within the community, it is possible to state that the only correct and robust choice that can actually be standardized would be the use of a new TLV to be encoded in the Options Header (Hop-by-Hop or Destination Option).

This approach is compliant with [RFC8200] indeed the Alternate Marking application to IPv6 involves the following operations:

- The source node is the only one that writes the Option Header to mark alternately the flow (for both Hop-by-Hop and Destination Option).

- In case of Hop-by-Hop Option Header carrying Alternate Marking bits, it is not inserted or deleted, but can be read by any node along the path. The intermediate nodes may be configured to support this Option or not. Anyway this does not impact the traffic since the measurement can be done only for the nodes configured to read the Option.

- In case of Destination Option Header carrying Alternate Marking bits, it is not processed, inserted, or deleted by any node along the path until the packet reaches the destination node. Note that, if there is also a Routing Header (RH), any visited destination in the route list can process the Option Header.

Hop-by-Hop Option Header is also useful to signal to routers on the path to process the Alternate Marking, anyway it is to be expected that some routers cannot process it unless explicitly configured.

The optimization of both implementation and scaling of the Alternate Marking Method is also considered and a way to identify flows is required. The Flow Monitoring Identification field (FlowMonID), as introduced in the next sections, goes in this direction and it is used to identify a monitored flow.

Note that the FlowMonID is different from the Flow Label field of the IPv6 Header ([RFC8200]). Flow Label is used for application service, like load-balancing/equal cost multi-path (LB/ECMP) and QoS. Instead, FlowMonID is only used to identify the monitored flow. The reuse of flow label field for identifying monitored flows is not considered since it may change the application intent and forwarding behaviour. Furthermore the flow label may be changed en route and this may also violate the measurement task. Those reasons make the definition of the FlowMonID necessary for IPv6. Flow Label and FlowMonID within the same packet have different scope, identify different flows, and associate different uses.

An important point that will also be discussed in this document is the the uniqueness of the FlowMonID and how to allow disambiguation of the FlowMonID in case of collision. [RFC6437] states that the Flow Label cannot be considered alone to avoid ambiguity since it could be accidentally or intentionally changed en route for compelling operational security reasons and this could also happen to the IP addresses that can change due to NAT. But the Alternate Marking is usually applied in a controlled domain, which would not have NAT and there is no security issue that would necessitate rewriting Flow Labels. So, for the purposes of this document, both IP addresses and Flow Label should not change in flight and, in some cases, they could be considered together with the FlowMonID for disambiguation.

3. Definition of the AltMark Option

The desired choice is to define a new TLV for the Options Extension Headers, carrying the data fields dedicated to the alternate marking method.

3.1. Data Fields Format

The following figure shows the data fields format for enhanced alternate marking TLV. This AltMark data is expected to be encapsulated in the IPv6 Options Headers (Hop-by-Hop or Destination Option).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |  Option Type  |  Opt Data Len |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              FlowMonID                |L|D|     Reserved      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- o Option Type: 8 bit identifier of the type of Option that needs to be allocated. Unrecognised Types MUST be ignored on receipt. For Hop-by-Hop Options Header or Destination Options Header, [RFC8200] defines how to encode the three high-order bits of the Option Type field. The two high-order bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type; for AltMark these two bits MUST be set to 00 (skip over this Option and continue processing the header). The third-highest-order bit specifies whether or not the Option Data can change en route to the packet’s final destination; for AltMark the value of this bit MUST be set to 0 (Option Data does not change en route).

- o Opt Data Len: The length of the Option Data Fields of this Option in bytes.

- o FlowMonID: 20 bits unsigned integer. The FlowMon identifier is described hereinafter.

- o L: Loss flag for Packet Loss Measurement as described hereinafter;

- o D: Delay flag for Single Packet Delay Measurement as described hereinafter;

- o Reserved: is reserved for future use. These bits MUST be set to zero on transmission and ignored on receipt.

4. Use of the AltMark Option

The AltMark Option is the best way to implement the Alternate Marking method and can be carried by the Hop-by-Hop Options header and the Destination Options header. In case of Destination Option, it is processed only by the source and destination nodes: the source node inserts and the destination node removes it. While, in case of Hop-by-Hop Option, it may be examined by any node along the path, if explicitly configured to do so. In this way an unrecognized Hop-by-Hop Option may be just ignored without impacting the traffic.

So it is important to highlight that the Option Layout can be used both as Destination Option and as Hop-by-Hop Option depending on the Use Cases and it is based on the chosen type of performance measurement. In general, it is needed to perform both end to end and hop by hop measurements, and the alternate marking methodology allows, by definition, both performance measurements. Anyway, in many cases the end-to-end measurement is not enough and it is required also the hop-by-hop measurement, so the most complete choice is the Hop-by-Hop Options Header.

IPv6, as specified in [RFC8200], allows nodes to optionally process Hop-by-Hop headers. Specifically 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. Also, it is 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.

The Hop-by-Hop Option defined in this document is designed to take advantage of the property of how Hop-by-Hop options are processed. Nodes that do not support this Option SHOULD ignore them. This can mean that, in this case, the performance measurement does not account for all links and nodes along a path.

Another application that can be mentioned is the presence of a Routing Header, in particular it is possible to consider SRv6. SRv6 leverages the Segment Routing header which consists of a new type of routing header. Like any other use case of IPv6, Hop-by-Hop and Destination Options are useable when SRv6 header is present. Because SRv6 is implemented through a Segment Routing Header (SRH), Destination Options before the Routing Header are processed by each destination in the route list, that means, in case of SRH, by every node that is an identity in the SR path.

In summary, it is possible to list the alternative possibilities:

- o Destination Option => measurement only by node in Destination Address.

- o Hop-by-Hop Option => every router on the path with feature enabled.

- o Destination Option + any Routing Header => every destination node in the route list.

In general, Hop-by-Hop and Destination Options are the most suitable ways to implement Alternate Marking.

It is worth mentioning that new Hop-by-Hop Options are not strongly recommended in [RFC7045] and [RFC8200], unless there is a clear justification to standardize it, because nodes may be configured to ignore the Options Header, drop or assign packets containing an Options Header to a slow processing path. In case of the AltMark data fields described in this document, the motivation to standardize a new Hop-by-Hop Option is that it is needed for OAM. An intermediate node can read it or not but this does not affect the packet behavior. The source node is the only one that writes the Hop-by-Hop Option to mark alternately the flow, so, the performance measurement can be done for those nodes configured to read this Option, while the others are simply not considered for the metrics.

In addition to the previous alternatives, for legacy network it is possible to mention a non-conventional application of the Destination Option for the hop by hop usage. [RFC8200] defines that the nodes along a path examine and process the Hop-by-Hop Options header only if Hop-by-Hop processing is explicitly configured. On the other hand, using the Destination Option for hop by hop action would cause worse performance than Hop-by-Hop. The only motivation for the hop by hop usage of Destination Options can be for compatibility reasons but in general it is not recommended.

5. Alternate Marking Method Operation

This section describes how the method operates. [RFC8321] introduces several alternatives but in this section the most applicable methods are reported and a new fied is introduced to facilitate the deployment and improve the scalability.

5.1. Packet Loss Measurement

The measurement of the packet loss is really straightforward. The packets of the flow are grouped into batches, and all the packets within a batch are marked by setting the L bit (Loss flag) to a same value. The source node can switch the value of the L bit between 0 and 1 after a fixed number of packets or according to a fixed timer, and this depends on the implementation. By counting the number of packets in each batch and comparing the values measured by different network nodes along the path, it is possible to measure the packet loss occurred in any single batch between any two nodes. Each batch represents a measurable entity unambiguously recognizable by all network nodes along the path.

Packets with different L values may get swapped at batch boundaries, and in this case, it is required that each marked packet can be assigned to the right batch by each router. It is important to mention that for the application of this method there are two elements to consider: the clock error between network nodes and the network delay. These can create offsets between the batches and out-of-order of the packets. There is the condition on timing aspects explained in [RFC8321] that must be satisfied and it takes into considerations the different causes of reordering such as clock error, network delay. The consequence is that it is necessary to define a waiting interval where to get stable counters and to avoid these issues. Usually the counters can be taken in the middle of the batch period to be sure to take still counters. In a few words this implies that the length of the batches MUST be chosen large enough so that the method is not affected by those factors.

```
   L bit=1   ----------+           +-----------+           +----------
                       |           |           |           |
   L bit=0             +-----------+           +-----------+
              Batch n        ...      Batch 3     Batch 2     Batch 1
            <---------> <---------> <---------> <---------> <--------->

                                Traffic Flow
            ===========================================================>
   L bit   ...1111111111 0000000000 11111111111 00000000000 111111111...
            ===========================================================>
```

Figure 1: Packet Loss Measurement and Single-Marking Methodology using L bit

5.2. Packet Delay Measurement

The same principle used to measure packet loss can be applied also to one-way delay measurement. Delay metrics MAY be calculated using the two possibilities:

1. Single-Marking Methodology: This approach uses only the L bit to calculate both packet loss and delay. In this case, the D flag MUST be set to zero on transmit and ignored by the monitoring points. The alternation of the values of the L bit can be used as a time reference to calculate the delay. Whenever the L bit changes and a new batch starts, a network node can store the timestamp of the first packet of the new batch, that timestamp can be compared with the timestamp of the first packet of the same batch on a second node to compute packet delay. Anyway this measurement is accurate only if no packet loss occurs and if there is no packet reordering at the edges of the batches. A different approach can also be considered and it is based on the concept of the mean delay. The mean delay for each batch is calculated by considering the average arrival time of the packets

for the relative batch. There are limitations also in this case indeed, each node needs to collect all the timestamps and calculate the average timestamp for each batch. In addition the information is limited to a mean value.

2. Double-Marking Methodology: This approach is more complete and uses the L bit only to calculate packet loss and the D bit (Delay flag) is fully dedicated to delay measurements. The idea is to use the first marking with the L bit to create the alternate flow and, within the batches identified by the L bit, a second marking is used to select the packets for measuring delay. The D bit creates a new set of marked packets that are fully identified over the network, so that a network node can store the timestamps of these packets; these timestamps can be compared with the timestamps of the same packets on a second node to compute packet delay values for each packet. The most efficient and robust mode is to select a single double-marked packet for each batch, in this way there is no time gap to consider between the double-marked packets to avoid their reorder. If a double-marked packet is lost, the delay measurement for the considered batch is simply discarded, but this is not a big problem because it is easy to recognize the problematic batch and skip the measurement just for that one. So in order to have more information about the delay and to overcome out-of-order issues this method is preferred.

```
L bit=1   ----------+           +-----------+           +----------
                    |           |           |           |
L bit=0             +-----------+           +-----------+

D bit=1         +          +          +          +            +
                |          |          |          |            |
D bit=0   ------+----------+----------+----------+------------+-----

                             Traffic Flow
         ===========================================================>
L bit   ...1111111111 0000000000 11111111111 00000000000 111111111...

D bit   ...0000010000 0000010000 00000100000 00001000000 000001000...
         ===========================================================>
```

Figure 2: Double-Marking Methodology using L bit and D bit

Similar to packet delay measurement (both for Single Marking and Double Marking), the method can also be used to measure the inter-arrival jitter.

5.3. Flow Monitoring Identification

The Flow Monitoring Identification (FlowMonID) is required for some general reasons:

- First, it helps to reduce the per node configuration. Otherwise, each node needs to configure an access-control list (ACL) for each of the monitored flows. Moreover, using a flow identifier allows a flexible granularity for the flow definition.

- Second, it simplifies the counters handling. Hardware processing of flow tuples (and ACL matching) is challenging and often incurs into performance issues, especially in tunnel interfaces.

- Third, it eases the data export encapsulation and correlation for the collectors.

The FlowMon identifier field is to uniquely identify a monitored flow within the measurement domain. The field is set at the source node. The FlowMonID can be uniformly assigned by the central controller or algorithmically generated by the source node. The latter approach cannot guarantee the uniqueness of FlowMonID but it may be preferred for local or private network, where the conflict probability is small due to the large FlowMonID space.

5.3.1. Uniqueness of FlowMonID

It is important to note that if the 20 bit FlowMonID is set independently and pseudo randomly there is a chance of collision. So, in some cases, FlowMonID could not be sufficient for uniqueness.

In general the probability of a flow identifier uniqueness correlates to the amount of entropy of the inputs. For instance, using the well-known birthday problem in probability theory, if the 20 bit FlowMonID is set independently and pseudo randomly without any additional input entropy, there is a 50% chance of collision for just 1206 flows. For a 32 bit identifier the 50% threshold jumps to 77,163 flows and so on. So, for more entropy, FlowMonID can either be combined with other identifying flow information in a packet (e.g. it is possible to consider the hashed 3-tuple Flow Label, Source and Destination addresses) or the FlowMonID size could be increased.

This issue is more visible when the FlowMonID is pseudo randomly generated by the source node and there needs to tag it with additional flow information to allow disambiguation. While, in case of a centralized controller, the controller should set FlowMonID by considering these aspects and instruct the nodes properly in order to guarantee its uniqueness.

5.4. Multipoint and Clustered Alternate Marking

The Alternate Marking method can also be extended to any kind of multipoint to multipoint paths, and the network clustering approach allows a flexible and optimized performance measurement, as described in [RFC8889].

The Cluster is the smallest identifiable subnetwork of the entire Network graph that still satisfies the condition that the number of packets that goes in is the same that goes out. With network clustering, it is possible to use the partition of the network into clusters at different levels in order to perform the needed degree of detail. So, for Multipoint Alternate Marking, FlowMonID can identify in general a multipoint-to-multipoint flow and not only a point-to-point flow.

5.5. Data Collection and Calculation

The nodes enabled to perform performance monitoring collect the value of the packet counters and timestamps. There are several alternatives to implement Data Collection and Calculation, but this is not specified in this document.

6. Security Considerations

This document aims to apply a method to perform measurements that does not directly affect Internet security nor applications that run on the Internet. However, implementation of this method must be mindful of security and privacy concerns.

There are two types of security concerns: potential harm caused by the measurements and potential harm to the measurements.

Harm caused by the measurement: Alternate Marking implies modifications on the fly to an Option Header of IPv6 packets by the source node but this must be performed in a way that does not alter the quality of service experienced by the packets and that preserves stability and performance of routers doing the measurements. The advantage of the Alternate Marking method is that the marking bits are the only information that is exchanged between the network nodes. Therefore, network reconnaissance through passive eavesdropping on data-plane traffic does not allow attackers to gain information about the network performance. Moreover, Alternate Marking should usually be applied in a controlled domain and this also helps to limit the problem.

Harm to the Measurement: Alternate Marking measurements could be harmed by routers altering the marking of the packets or by an attacker injecting artificial traffic. Since the measurement itself may be affected by network nodes along the path intentionally altering the value of the marking bits of IPv6 packets, the Alternate Marking should be applied in the context of a controlled domain, where the network nodes are locally administered and this type of attack can be avoided. Indeed the source and destination addresses are within the controlled domain and therefore it is unlikely subject to hijacking of packets, because it is possible to filter external packets at the domain boundaries. In addition, an attacker cannot gain information about network performance from a single monitoring point; it must use synchronized monitoring points at multiple points on the path, because they have to do the same kind of measurement and aggregation as Alternate Marking requires.

The privacy concerns of network measurement are limited because the method only relies on information contained in the Option Header without any release of user data. Although information in the Option Header is metadata that can be used to compromise the privacy of users, the limited marking technique seems unlikely to substantially increase the existing privacy risks from header or encapsulation metadata.

The Alternate Marking application described in this document relies on an time synchronization protocol. Thus, by attacking the time protocol, an attacker can potentially compromise the integrity of the measurement. A detailed discussion about the threats against time protocols and how to mitigate them is presented in [RFC7384].

7. IANA Considerations

The Option Type should be assigned in IANA's "Destination Options and Hop-by-Hop Options" registry.

This draft requests the following IPv6 Option Type assignments from the Destination Options and Hop-by-Hop Options sub-registry of Internet Protocol Version 6 (IPv6) Parameters (https://www.iana.org/assignments/ipv6-parameters/).

Hex Value	Binary Value act chg rest	Description	Reference
TBD	00 0 tbd	AltMark	[This draft]

8. Acknowledgements

The authors would like to thank Bob Hinden, Ole Troan, Tom Herbert, Stefano Previdi, Brian Carpenter, Eric Vyncke, Ron Bonica for the precious comments and suggestions.

9. References

9.1. 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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

9.2. Informative References

[I-D.fioccola-v6ops-ipv6-alt-mark] Fioccola, G., Velde, G., Cociglio, M., and P. Muley, "IPv6 Performance Measurement with Alternate Marking Method", draft-fioccola-v6ops-ipv6-alt-mark-01 (work in progress), June 2018.

[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>.

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

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

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <https://www.rfc-editor.org/info/rfc7384>.

[RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <https://www.rfc-editor.org/info/rfc8321>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

[RFC8889] Fioccola, G., Ed., Cociglio, M., Sapio, A., and R. Sisto, "Multipoint Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8889, DOI 10.17487/RFC8889, August 2020, <https://www.rfc-editor.org/info/rfc8889>.

Authors' Addresses

Giuseppe Fioccola
Huawei
Riesstrasse, 25
Munich 80992
Germany

Email: giuseppe.fioccola@huawei.com

Tianran Zhou
Huawei
156 Beiqing Rd.
Beijing 100095
China

Email: zhoutianran@huawei.com

Mauro Cociglio
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: mauro.cociglio@telecomitalia.it

Fengwei Qin
China Mobile
32 Xuanwumenxi Ave.
Beijing 100032
China

Email: qinfengwei@chinamobile.com

Ran Pang
China Unicom
9 Shouti South Rd.
Beijing 100089
China

Email: pangran@chinaunicom.cn

Network Working Group R. Hinden
Internet-Draft Check Point Software
Intended status: Experimental G. Fairhurst
Expires: 26 April 2021 University of Aberdeen
23 October 2020

IPv6 Minimum Path MTU Hop-by-Hop Option
draft-ietf-6man-mtu-option-04

Abstract

This document specifies a new Hop-by-Hop IPv6 option that is used to record the minimum Path MTU along the forward path between a source host to a destination host. This collects a minimum Path MTU recorded along the path to the destination. The value can then be communicated back to the source using the return Path MTU field in the option.

This Hop-by-Hop option is intended to be used in environments like Data Centers and on paths between Data Centers, to allow them to better take advantage of paths able to support a large Path MTU. The method could also be useful in other environments, including the general Internet.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This draft proposes a new IPv6 Hop-by-Hop Option to be used to record the minimum of the Maximum Transmission Unit (MTU) along the forward path between the source and destination hosts. The source host creates a packet with this option and fills the Min-PMTU field with the value of the MTU for the outbound link that will be used to forward the packet towards the destination host.

At each subsequent hop where the option is processed, the router compares the value of the Min-PMTU Field in the option and the MTU of its outgoing link. If the MTU of the link is less than the Min-PMTU, it rewrites the value in the option data with the smaller value. When the packet arrives at the destination host, the host can send the value of the minimum reported MTU for the path back to the source host using the Rtn-PMTU field in the option. The source host can then use this value as an input to the method that sets the Path MTU (PMTU) used by upper layer protocols.

1.1. Example Operation

The figure below illustrates the operation of the method. In this case, the path between the source and destination hosts comprises three links, the sender has a link MTU of size MTU-S, the link between routers R1 and R2 has an MTU of size 9000 bytes, and the final link to the destination has an MTU of size MTU-D.

```
+--------+         +----+         +----+         +-------+
|        |         |    |         |    |         |       |
| Sender +---------+ R1 +---------+ R2 +-------- + Dest. |
|        |         |    |         |    |         |       |
+--------+  MTU-S  +----+  9000B +----+  MTU-D  +-------+
```

Three scenarios are described:

* Scenario 1, considers all links to have an 9000 byte MTU and the method is supported by both routers. The PMTU is therefore 9000 bytes.

* Scenario 2, considers the link to the destination host (MTU-D) to have an MTU of 1500 bytes. This is the smallest MTU, router R2 updates the Min-PMTU to 1500 bytes and the method correctly updates the PMTU to 1500 bytes. Had there been another smaller MTU at a link further along the path that also supports the method, the lower MTU would also have been detected.

* Scenario 3, considers the case where the router preceding the smallest link (R2) does not support the method, and the link to the destination host (MTU-D) has an MTU of 1500 bytes. Therefore, router R2 does not update the Min-PMTU to 1500 bytes. The method then fails to detect the actual PMTU.

In Scenarios 2 and 3, a lower PMTU would also fail to be detected in the case where PMTUD had been used and an ICMPv6 Packet to Big (PTB) message had not been delivered to the sender [RFC8201].

These scenarios are summarized in the table below.

	MTU-S	MTU-D	R1	R2	Rec PMTU	Note
1	9000B	9000B	H	H	9000 B	Endpoints attempt to use an 9000 B PMTU.
2	9000B	1500B	H	H	1500 B	Endpoints attempt to use a 1500 B PMTU.
3	9000B	1500B	H	-	9000 B	Endpoints attempt to use an 9000 B PMTU, but need to implement a method to fall back to discover and use a 1500 B PMTU.

1.2. Use of the IPv6 Hop-by-Hop Options Header

IPv6 as specified in [RFC8200] allows nodes to optionally process Hop-by-Hop headers. Specifically from Section 4:

* 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.

The Hop-by-Hop Option defined in this document is designed to take advantage of this property of how Hop-by-Hop options are processed. Nodes that do not support this Option SHOULD ignore them. This can mean that the Min-PMTU value does not account for all links along a path.

2. Motivation and Problem Solved

The current state of Path MTU Discovery on the Internet is problematic. The mechanisms defined in [RFC8201] are known to not work well in all environments. This fails to work in various cases, including when nodes in the middle of the network do not send ICMP PTB messages, or rate-limited messages to the point of not making them a useful mechanism, or do not have a return path to the source host.

This results in many transport connections being configured to use smaller packets (e.g., 1280 bytes) by default and makes it difficult to take advantage of paths with a larger PMTU where they do exist. Applications that can gain benefit from sending large packets are forced to use IPv6 Fragmentation [RFC8200], which can reduce the reliability of Internet communication [RFC8900].

Transport encapsulations and network-layer tunnels further reduce the the payload size available for a transport to use. Also, some use-cases increase packet overhead, for example, Network Virtualization Using Generic Routing Encapsulation (NVGRE) [RFC7637] encapsulates L2 packets in an outer IP header and does not allow IP Fragmentation.

Sending small packets can limit performance, e.g., when packet processing is limited by the packet rate. The potential of multi-gigabit Ethernet will not be realized if the packet size is limited to 1280 bytes, because this exceeds the packet per second rate that most nodes can process. For example, the packet per second rate required to reach wire speed on a 10G Ethernet link with 1280 byte packets is about 977K packets per second (pps), vs. 139K pps for 9000 byte packets. A significant difference.

The purpose of the this draft is to improve the situation by defining a mechanism that does not rely on reception of ICMPv6 Packet Too Big messages from nodes in the middle of the network. Instead, this provides information to the destination host about the minimum Path MTU, and sends this information back to the source host. This is expected to work better than the current RFC8201-based mechanisms.

3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

4. Applicability Statements

This Hop-by-Hop Option header is intended to be used in environments such as Data Centers and on paths between Data Centers, to allow a host to better take advantage of a path that is able to support a large PMTU.

The design of the option is sufficiently simple that it could be executed on a router's fast path. A strong pull from router vendors customers will be required to create critical mass for this to happen. This could initially be the case for connections within and between Data Centers.

The method could also be useful in other environments, including the general Internet, if and when this Hop-by-Hop Option is supported on these paths.

5. IPv6 Minimum Path MTU Hop-by-Hop Option

The Minimum Path MTU Hop-by-Hop Option has the following format:

```
 Option    Option    Option
  Type    Data Len   Data
+--------+--------+--------+--------+---------+-------+-+
|BBCTTTTT|00000100|     Min-PMTU    |     Rtn-PMTU    |R|
+--------+--------+--------+--------+---------+-------+-+
```

```
  Option Type (see Section 4.2 of [RFC8200]):

  BB     00   Skip over this option and continue processing.

  C       1   Option data can change en route to the packet's final
              destination.

  TTTTT 10000 Option Type assigned from IANA [IANA-HBH].

  Length:  4  The size of the each value field in Option Data
              field supports PMTU values from 0 to 65,535 octets.

  Min-PMTU: n 16-bits.  The minimum MTU recorded along the path
              in octets, reflecting the smallest link MTU that
              the packet experienced along the path.
              A value less than the IPv6 minimum link
              MTU [RFC8200] should be ignored.

  Rtn-PMTU: n 15-bits.  The returned Path MTU field, carrying the 15
              most significant bits of the latest received Min-PMTU
              field for the forward path.  The value zero means that
              no Reported MTU is being returned.

  R        n  1-bit.  R-Flag.   Set by the source to signal that
              the destination host should include the received
              Rtn-PMTU field updated by the reported Min-PMTU value.
```

NOTE: The encoding of the final two octets (Rtn-PMTU and R-Flag) could be implemented by a mask of the latest received Min-PMTU value with 0xFFFE, discarding the right-most bit and then performing a logical 'OR' with the R-Flag value of the sender.

6. Router, Host, and Transport Behaviors

6.1. Router Behavior

Routers that are not configured to support Hop-by-Hop Options SHOULD ignore this option and SHOULD forward the packet.

Routers that support Hop-by-Hop Options, but that are not configured to support this option SHOULD ignore the option and SHOULD forward the packet.

Routers that recognize this option SHOULD compare the value of the Min-PMTU field with the MTU configured for the outgoing link. If the MTU of the outgoing link is less than the Min-PMTU, the router rewrites the Min-PMTU in the Option to use the smaller value.

A router MUST ignore and MUST NOT change the Rtn-PMTU field or the R-Flag in the option.

Discussion:

* The design of this option makes it feasible to be implemented within the fast path of a router, because the processing requirements are minimal.

6.2. Host Behavior

When requested to send an IPv6 packet with the Minimum Path MTU option, the source host includes the option in an outgoing packet. The source host SHOULD fill the Min-PMTU field with the MTU configured for the link over which it will send the packet on the next hop towards the destination host. If this value is not updated, the field MUST be set to zero.

The source host SHOULD set the Rtn-PMTU field to the cached value of the reported Min-PMTU value for the flow (see Section 6.3.3). If this value is not set, for example, because there is no cached reported Min-PMTU value, the field MUST be set to zero.

The source host MAY request the destination host to return the reported Min-PMTU value by setting the R-Flag in the option of an outgoing packet.

6.3. Transport Behavior

6.3.1. Including the Option in an Outgoing Packet

The upper layer protocol can request the Minimum Path MTU option is included in an outgoing IPv6 packet. This option does not need to be included in all packets belonging to a flow. A transport protocol (or upper layer protocol) can include this option only on specific packets used to test the path.

When it includes the option, the host supplies the previously cached value of the received Minimum Path MTU for the flow to set the Rtn-PMTU field (see Section 6.3.3). If a valid cached received Minimum Path MTU is not available, the Rtn-PMTU field value MUST be set to zero.

The source host MAY request the destination host to send a packet carrying the option by setting the R-Flag. The R-Flag SHOULD NOT be set when the Minimum Path MTU Option was sent solely to feedback the return Path MTU.

NOTE: Including this option in a large packet (e.g., one larger than the present PMTU) is not likely to be useful, since the large packet would itself be dropped by any link along the path with a smaller MTU, preventing the Min-PMTU information from reaching the destination host.

Discussion:

* In the case of TCP, the option could be included in packets carrying a SYN segment as part of the connection set up, or can periodically be sent in packets carrying other segments. Including this packet in a SYN could increase the probability that the SYN segment is lost when routers on the path drop packets with this option (see Section 6.3.5). NOTE: A TCP connection can also negotiate the Maximum Segment Size (MSS), which acts as an upper limit to the packet size that can be sent by a TCP sender.

* The use with datagram transport protocols (e.g., UDP) is harder to characterize because applications using datagram transports range from very short-lived (low data-volume applications) exchanges, to longer (bulk) exchanges of packets between the source and destination hosts [RFC8085].

* Simple-exchange protocols (i.e., low data-volume applications [RFC8085] that only send one or a few packets per transaction, might assume that the PMTU is symmetrical. That is, the PMTU is the same in both directions, or at least not smaller for the return path. This optimization does not hold when the paths are not symmetric.

* The use of this option with DNS and DNSSEC over UDP ought to work for paths where the PMTU is symmetric. The DNS server will learn the PMTU from the DNS query messages. If the Rtn-PMTU value is smaller, then a large DNSSEC response might be dropped and the known problems with PMTUD will then occur. DNS and DNSSEC over transport protocols that can carry the PMTU ought to work.

* Applications that use Anycast should include this option in all packets, because the actual destination host will vary due to the nature of Anycast.

6.3.2. Validation by the Upper Layer Protocol

An upper layer protocol (e.g., transport endpoint) using this option needs to provide protection from data injection attacks by off-path devices [RFC8085]. This requires a method to assure that the information in the Option Data is provided by a node on the path. For example, a TCP connection or UDP application that maintains the related state and uses a randomized ephemeral port would provide this basic validation to protect from off-path data injection. IPsec [RFC4301] and TLS [RFC8446] provide greater assurance.

The Upper Layer discards any received packet when the packet validation fails. When packet validation fails, the Upper Layer MUST also discard the associated Option Data from the minimum Path MTU option without further processing.

6.3.3. Receiving the Option

An upper layer protocol that receives a Minimum Path MTU Option included with a valid packet caches the value of the last received Min-PMTU. This value is specific to the instance of the upper layer protocol (i.e., matching the IPv6 flow ID, port-fields in UDP or the SPI in IPsec [RFC4301], etc), not to the pair of source and destination addresses, because network devices can make forwarding decisions that impact the PMTU of a flow based on the presence and value of the packet’s upper layer fields.

For a connection-oriented upper layer protocol, caching of the received Min-PMTU could be implemented by saving the value in the connection context at the transport layer. A connection-less upper layer (e.g., one using UDP), requires the upper layer protocol to cache the value for each flow it uses.

A destination host that receives a Minimum Path MTU Option with the R-Flag SHOULD include the Minimum Path MTU option in the next outgoing IPv6 packet for the corresponding flow.

A simple mechanism could only include this option (with the Rtn-PMTU field set) the first time this option is received or when it notifies a change in the Minimum Path MTU. This limits the number of packets including the option packets that are sent. However, this does not provide robustness to packet loss or recovery after a sender looses state.

Path characteristics can change and the actual PMTU could increase or decrease over time. For instance, following a path change when packets are then forwarded over a link with a different MTU than that previously used. To bound the delay in discovering a change in the actual PMTU, a sender with a link MTU larger than the current PMTU SHOULD periodically send the Minimum Path MTU Option with the R-bit set. DPLPMTUD provides recommendations concerning how this could be implemented (see Section 5.3 of [RFC8899]). Since the option consumes less capacity than a full-sized probe packet, there can be advantage in using this to detect a change in the path characteristics.

Discussion:

* Some upper layer protocols send packets less frequently than packets that the host receives packets. This provides less frequent feedback of the received Rtn-PMTU value. However, a host always sends the most recent Rtn-PMTU value.

6.3.4. Using the Rtn-PMTU Field

The Rtn-PMTU field provides an indication of the PMTU from on-path routers. It does not necessarily reflect the actual PMTU between the sender and destination. Care therefore needs to be exercised in using the Rtn-PMTU value. Specifically:

* The actual PMTU can be lower than the Rtn-PMTU value because Min-PMTU field was not updated by a router on the path that did not process the option.

* The actual PMTU may be lower than the Rtn-PMTU value because the there is a layer 2 device with a lower MTU that does not perform IPv6 forwarding.

* The actual PMTU may be larger than the Rtn-PMTU value because of a corrupted, delayed or mis-ordered response. A source host SHOULD ignore a Rtn-PMTU value larger than the MTU configured for the outgoing link.

Using the method has the potential to complete discovery of the correct value in a single round trip time, even over paths that have successive links each configured with a lower MTU.

To avoid unintentional dropping of packets that exceed the actual PMTU (e.g., Scenario 3 in Section 1.1), the source host can delay increasing the PMTU until a probe packet with the size of the Rtn-PMTU value has been successfully acknowledged by the upper layer, confirming that the path supports the larger PMTU. This probing increases robustness, but adds one additional path round trip time before the PMTU is updated. This use resembles that of PTB messages in section 4.6 of DPLPMTUD [RFC8899] (with the important difference that a PTB message can only seek to lower the PMTU, whereas this option could trigger a probe packet to seek to increase the PMTU.)

Section 5.2 of [RFC8201] provides guidance on the caching of PMTU information and also the relation to IPv6 flow labels. Implementations should consider the impact of Equal Cost Multipath (ECMP) [RFC6438]. Specifically, whether a PMTU ought be maintained for each transport endpoint, or for each network address.

6.3.5. Detection of Dropping Packets that include the Option

There is evidence that some middleboxes drop packets that include Hop-by-Hop options. For example, a firewall might drop a packet that carries an unknown extension header or option. This practice is expected to decrease as an option becomes more widely used. It could result in generation of an ICMPv6 message indicating the problem. This could be used to (temporarily) suspend use of this option.

A middlebox that silently discards a packet with this option results in dropping of any packet using the option. This dropping be avoided by appropriate configuration in a controlled environment, such as within a data centre, but needs to be considered for Internet usage. Section 6.2 recommends that this option is not used on packets where loss might adversely impact performance.

7. IANA Considerations

No IANA actions are requested in this document.

IANA has assigned and registered a new IPv6 Hop-by-Hop Option type from the "Destination Options and Hop-by-Hop Options" registry [IANA-HBH]. This assignment is shown in Section 5.

8. Security Considerations

This section discusses the security considerations. It first reviews host processing when receiving this option at the network layer. It then considers two ways in which the Option Data can be processed, followed by two approaches for using the Option Data. Finally, it discusses middlebox implications related to use in the general Internet.

8.1. Network Layer Host Processing

A malicious attacker can forge a packet directed at a host that carries the minimum Path MTU option. By design, the fields of this IP option can be modified by the network.

Reception of this packet will incur receive processing as the network stack parses the packet before the packet is delivered to the upper layer protocol. This network layer option processing is normally completed before any upper layer protocol delivery checks are performed.

The network layer does not normally have sufficient information to validate that the packet carrying an option originated from the destination (or an on-path node). It also does not typically have sufficient context to demultiplex the packet to identify the related transport flow. This can mean that any changes resulting from reception of the option apply to all flows between a pair of endpoints.

These considerations are no different to other uses of Hop-by-Hop options, and this is the use case for PMTUD. The following section describes a mitigation for this attack.

8.2. Validating use of the Option Data

Transport protocols should be designed to provide protection from data injection attacks by off-path devices and mechanisms should be described in the Security Considerations for each transport specification (see Section 5.1 of the UDP Guidelines [RFC8085]). For example, a TCP or UDP application that maintains the related state and uses a randomized ephemeral port would provide basic protection. TLS [RFC8446] or IPsec [RFC4301] provide cryptographic authentication. An upper layer protocol that validates each received packet discards any packet when this validation fails. In this case, the host MUST also discard the associated Option Data from the minimum Path MTU option without further processing (Section 6.3).

A network node on the path has visibility of all packets it forwards. By observing the network packet payload, the node might be able to construct a packet that might be validated by the destination host. Such a node would also be able to drop or limit the flow in other ways that could be potentially more disruptive. Authenticating the packet, for example, using IPsec [RFC4301] or TLS [RFC8446] mitigates this attack.

8.3. Direct use of the Rtn-PMTU Value

The simplest way to utilize the Rtn-PMTU value is to directly use this to update the PMTU. This approach results in a set of security issues when the option carries malicious data:

* A direct update of the PMTU using the Rtn-PMTU value could result in an attacker inflating or reducing the size of the host PMTU for the destination. Forcing a reduction in the PMTU can decrease the efficiency of network use, might increase the number of packets/ fragments required to send the same volume of payload data, and prevents sending an unfragmented datagram larger than the PMTU. Increasing the PMTU can result in black-holing (see Section 1.1 of [RFC8899]) when the source sends packets larger than the actual PMTU. This persists until the PMTU is next updated.

* The method can be used to solicit a response from the destination host. A malicious attacker could forge a packet that cause the sender to add the option to a packet sent to the source. A forged value of Rtn-PMTU in the Option Data might also impact the remote endpoint, as described in the previous bullet. This persists until a valid minimum Path MTU option is received. This attack could be mitigated by limiting the sending of the minimum Path MTU option in reply to incoming packets that carry the option.

8.4. Using the Rtn-PMTU Value as a Hint for Probing

Another way to utilize the Rtn-PMTU value is to indirectly trigger a probe to determine if the path supports a PMTU of size Rtn-PMTU. This approach needs context for the flow, and hence assumes an upper layer protocol that validates the packet that carries the option Section 8.2. This is the case when used in combination with DPLPMTUD [RFC8899]. A set of security considerations result when an option carries malicious data:

* If the forged packet carries a validated option with a non-zero Rtn-PMTU field, the upper layer protocol could utilize the information in the Rtn-PMTU field. A Rtn-PMTU larger than the current PMTU can trigger a probe for a new size.

* If the forged packet carries a non-zero Min-PMTU field, the upper layer protocol would change the cached information about the path from the source. The cached information at the destination host will be overwritten when the host receives another packet that includes a minimum Path MTU option corresponding to the flow.

* Processing of the option could cause a destination host to add the minimum Path MTU option to a packet sent to the source host. This option will carry a Rtn-PMTU value that could have been updated by the forged packet. The impact of the source host receiving this resembles that discussed previously.

8.5. Impact of Middleboxes

There is evidence that some middleboxes drop packets that include Hop-by-Hop options. For example, a firewall might drop a packet that carries an unknown extension header or option. This practice is expected to decrease as the option becomes more widely used. Methods to address this are discussed in Section 6.3.5.

When a forged packet cause a packet to be sent including the minimum Path MTU option, and the return path does not forward packets with this option, the packet will be dropped Section 6.3.5. This attack is mitigated by validating the option data before use and by limiting the rate of responses generated. An upper layer could further mitigate the impact by responding to a R-Flag by including the option in a packet that does not carry application data.

9. Acknowledgments

A somewhat similar mechanism was proposed for IPv4 in 1988 in [RFC1063] by Jeff Mogul, C. Kent, Craig Partridge, and Keith McCloghire. It was later obsoleted in 1990 by [RFC1191] the current deployed approach to Path MTU Discovery.

Helpful comments were received from Tom Herbert, Tom Jones, Fred Templin, Ole Troan, [Your name here], and other members of the 6MAN working group.

10. Change log [RFC Editor: Please remove]

draft-ietf-6man-mtu-option-04, 2020-Oct-23

* Fixes for typos.

draft-ietf-6man-mtu-option-03, 2020-Sept-14

* Rewrite to make text and terminology more consistent.

* Added the notion of validating the packet before use of the HBH option data.
* Method aligned with the way common APIs send/receive HBH option data.
* Added reference to DPLPMTUD and clarified upper layer usage.
* Completed security considerations section.

draft-ietf-6man-mtu-option-02, 2020-March-9

* Editorial changes to make text and terminology more consistent.
* Added reference to DPLPMTUD.

draft-ietf-6man-mtu-option-01, 2019-September-13

* Changes to show IANA assigned code point.
* Editorial changes to make text and terminology more consistent.
* Added a reference to RFC8200 in Section 2 and a reference to RFC6438 in Section 6.3.

draft-ietf-6man-mtu-option-00, 2019-August-9

* First 6man w.g. draft version.
* Changes to request IANA allocation of code point.
* Editorial changes.

draft-hinden-6man-mtu-option-02, 2019-July-5

* Changed option format to also include the Returned PMTU value and Return flag and made related text changes in Section 6.2 to describe this behavior.
* ICMP Packet Too Big messages are no longer used for feedback to the source host.
* Added to Acknowledgements Section that a similar mechanism was proposed for IPv4 in 1988 in [RFC1063].
* Editorial changes.

draft-hinden-6man-mtu-option-01, 2019-March-05

* Changed requested status from Standards Track to Experimental to allow use of experimental option type (11110) to allow for experimentation. Removed request for IANA Option assignment.
* Added Section 2 "Motivation and Problem Solved" section to better describe what the purpose of this document is.
* Added appendix describing planned experiments and how the results will be measured.
* Editorial changes.

draft-hinden-6man-mtu-option-00, 2018-Oct-16

* Initial draft.

11. References

11.1. Normative References

[IANA-HBH] "Destination Options and Hop-by-Hop Options", <https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

11.2. Informative References

[RFC1063] Mogul, J., Kent, C., Partridge, C., and K. McCloghrie, "IP MTU discovery options", RFC 1063, DOI 10.17487/RFC1063, July 1988, <https://www.rfc-editor.org/info/rfc1063>.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://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, <https://www.rfc-editor.org/info/rfc4301>.

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

[RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <https://www.rfc-editor.org/info/rfc7637>.

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

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8899] Fairhurst, G., Jones, T., Tüxen, M., Rüngeler, I., and T. Völker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <https://www.rfc-editor.org/info/rfc8899>.

[RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <https://www.rfc-editor.org/info/rfc8900>.

Authors' Addresses

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, CA 94070
United States of America

Email: bob.hinden@gmail.com

Godred Fairhurst
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen
AB24 3UE
United Kingdom

Email: gorry@erg.abdn.ac.uk

IPv6 Maintenance (6man) Working Group   F. Gont
Internet-Draft   SI6 Networks
Updates: 4861, 4862 (if approved)   J. Zorz
Intended status: Standards Track   6connect
Expires: July 23, 2021   R. Patterson
                                                                  Sky UK
                                                        January 19, 2021

Improving the Robustness of Stateless Address Autoconfiguration (SLAAC) to Flash Renumbering Events

draft-ietf-6man-slaac-renum-02

Abstract

In renumbering scenarios where an IPv6 prefix suddenly becomes invalid, hosts on the local network will continue using stale prefixes for an unacceptably long period of time, thus resulting in connectivity problems. This document improves the reaction of IPv6 Stateless Address Autoconfiguration to such renumbering scenarios.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

IPv6 network renumbering is expected to take place in a planned manner, with old/stale prefixes being phased-out via reduced prefix lifetimes while new prefixes (with normal lifetimes) are introduced. However, there are a number of scenarios that may lead to the so-called "flash-renumbering" events, where the prefix being employed on a network suddenly becomes invalid and replaced by a new prefix [I-D.ietf-v6ops-slaac-renum]. In such scenarios, hosts on the local network will continue using stale prefixes for an unacceptably long period of time, thus resulting in connectivity problems. [I-D.ietf-v6ops-slaac-renum] discusses this problem in detail.

In some scenarios, the local router producing the network renumbering event may try to deprecate the currently-employed prefixes (thus explicitly signaling the network about the renumbering event), whereas in other scenarios it may be unaware about the renumbering event and thus unable signal hosts about it.

From the perspective of a Stateless Address Autoconfiguration (SLAAC) host, there are two different (but related) problems to be solved:

o Avoiding the use of stale addresses for new communication instances

o Performing "garbage collection" for the stale prefixes (and related network configuration information)

Clearly, if a host has both working and stale addresses, it is paramount that it employs working addresses for new communication instances. Additionally, a host should also perform garbage collection for the stale prefixes/addresses, since they not only tie system resources, but also prevent communication with the new "owners" of the stale prefixes.

2. Terminology

The term "globally reachable" is used in this document as defined in [RFC8190].

The term "Global Unicast Address" (or its acronym "GUA") is used throughout this document to refer to "globally reachable" [RFC8190] addresses. That is, when used throughout this document, GUAs do NOT include Unique Local Addresses (ULAs) [RFC4193]. Similarly, the term "Global Unicast prefix" (or "GUA prefix") is employed throughout this document to refer to network prefixes that specify GUAs, and does NOT include the ULA prefix (FC00::/7) [RFC4193].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. SLAAC reaction to Flash-renumbering Events

As noted in Section 1, in some scenarios the router triggering the renumbering event may be able to explicitly signal the network about this event, while in other scenarios the renumbered hosts may need to infer a renumbering event is taking place. The following subsections analyze specific considerations for each of these scenarios.

3.1. Renumbering without Explicit Signaling

In the absence of explicit signalling from SLAAC routers (such as sending Prefix Information Options (PIOs) with small lifetimes to deprecate the stale prefixes), stale prefixes will remain preferred and valid according to the Preferred Lifetime and Valid Lifetime values (respectively) of the last received PIO. IPv6 SLAAC employs the following default values for PIOs:

- o Preferred Lifetime (AdvPreferredLifetime): 604800 seconds (7 days)

- o Valid Lifetime (AdvValidLifetime): 2592000 seconds (30 days)

This means that, in the absence of explicit signaling by a SLAAC router to deprecate a prefix, it will take a host 7 days (one week) to deprecate the corresponding addresses, and 30 days (one month) to eventually remove any addresses configured for the stale prefix. Clearly, for any practical purposes, employing such long default values is the equivalent of not using any timers at all, since taking 7 days or 30 days (respectively) to recover from a network problem is simply unacceptable.

Use of more appropriate timers in Router Advertisement messages can help limit the amount of time that hosts will maintain stale configuration information. Additionally, hosts are normally in a position to infer that a prefix has become stale -- for example, if a given router ceases to advertise an existing prefix and at the same time starts to advertise a new prefix.

Section 4.1.1 recommends the use of more appropriate default lifetimes for PIOs, while Section 4.5 specifies a local policy that SLAAC hosts can implement to heuristically infer that network configuration information has changed, such that stale configuration information can be phased out.

3.2. Renumbering with Explicit Signaling

In scenarios where a local router is aware about the renumbering event, it may try to phase out the stale network configuration information. In these scenarios, there are two aspects to be considered:

- o The amount of time during which the router should continue trying to deprecate the stale network configuration information

- o The ability of SLAAC hosts to phase out stale configuration in a timelier manner.

In the absence of Router Advertisements (RAs) that include PIOs that would reduce the Valid Lifetime and Preferred Lifetime of a prefix, hosts would normally employ the lifetime values from PIO options of the last received RA messages. Since the network could be partitioned for an arbitrarily long period of time, a router would need to try to deprecate a prefix for the amount of time employed for the "Preferred Lifetime", and try to invalidate the prefix for the amount of time employed for the "Valid Lifetime" (see Section 12 of [RFC4861]).

NOTE:
Once the number of seconds in the original "Preferred Lifetime" have elapsed, all hosts would have deprecated the corresponding addresses anyway, while once the number of seconds in the "Valid Lifetime" have elapsed, the corresponding addresses would be invalidated and removed.

Thus, use of more appropriate default lifetimes for PIOs, as proposed in Section 4.1.1, would reduce the amount of time a stale prefix would need to be announced as such by a router in order to make sure that it is deprecated/invalidated.

In scenarios where a router has positive knowledge that a prefix has become invalid and thus could signal this condition to local hosts, the current specifications will prevent SLAAC hosts from fully recovering from such stale information. Item "e)" of Section 5.5.3 of [RFC4862] specifies that an RA may never reduce the "RemainingLifetime" to less than two hours. Additionally, if the RemainingLifetime of an address is smaller than 2 hours, then a Valid Lifetime smaller than 2 hours will be ignored. The inability to invalidate a stale prefix would prevent communication with the new "owners" of the stale prefix, and thus is highly undesirable. On the other hand, the Preferred Lifetime of an address *can* be reduced to any value to avoid the use of a stale prefix for new communications.

Section 4.2 updates [RFC4862] such that this restriction in removed, and hosts react to the advertised "Valid Lifetime" (even if it is smaller than 2 hours).

Finally, Section 4.3 recommends that routers disseminate network configuration information when a network interface is initialized, such that possibly new configuration information propagates in a timelier manner.

4. Improvements to Stateless Address Autoconfiguration (SLAAC)

The following subsections update [RFC4861] and [RFC4862], such that the problem discussed in this document is mitigated. The aforementioned updates are mostly orthogonal, and mitigate different aspects of SLAAC that prevent a timely reaction to flash renumbering events.

- Reduce the default Valid Lifetime and Preferred Lifetime of PIOs (Section 4.1.1):
  This helps limit the amount of time a host will employ stale information, and also limits the amount of time a router needs to try to obsolete stale information.

- Honor PIOs with small Valid Lifetimes (Section 4.2):
  This allows routers to invalidate stale prefixes, since otherwise [RFC4861] prevents hosts from honoring PIOs with a Valid Lifetime smaller than two hours.

- Recommend routers to retransmit configuration information upon interface initialization/reinitialization (Section 4.3):
  This helps spread the new information in a timelier manner, and also deprecate stale information via host-side heuristics (see Section 4.5).

- Recommend routers to always send all options (i.e. the complete configuration information) in RA messages, and in the smallest possible number of packets (Section 4.4):
  This helps propagate the same information to all hosts, and also allows hosts to better infer that information missing in RA messages has become stale (see Section 4.5).

- Infer stale network configuration information from received RAs (Section 4.5):
  This allows hosts to deprecate stale network configuration information, even in the absence of explicit signaling.

4.1. More Appropriate Lifetime Values

4.1.1. Router Configuration Variables

The default values of the Preferred Lifetime and the Valid Lifetime of PIOs are updated as follows:

AdvPreferredLifetime: max(AdvDefaultLifetime, 3 * MaxRtrAdvInterval)

AdvValidLifetime: 2 * AdvPreferredLifetime

where:

AdvPreferredLifetime:
Value to be placed in the "Preferred Lifetime" field of the PIO.

AdvValidLifetime:
Value to be placed in the "Valid Lifetime" field of the PIO.

AdvDefaultLifetime:
Value to be placed in the "Router Lifetime" field of the Router Advertisement message that will carry the PIO.

max():
A function that computes the maximum of its arguments.

NOTE:
[RFC4861] specifies the default value of MaxRtrAdvInterval as 600 seconds, and the default value of AdvDefaultLifetime as 3 * MaxRtrAdvInterval. Therefore, when employing default values for MaxRtrAdvInterval and AdvDefaultLifetime, the default values of AdvPreferredLifetime and AdvValidLifetime become 1800 seconds (30 minutes) and 3600 seconds (1 one hour), respectively. We note that when implementing BCP202 [RFC7772], AdvDefaultLifetime will typically be in the range of 45-90 minutes, and therefore the default value of AdvPreferredLifetime will be in the range 45-90 minutes, while the default value of AdvValidLifetime will be in the range of 90-180 minutes.

RATIONALE:

* The default values of the PIO lifetimes should be such that, under normal circumstances (including some packet loss), the associated timers are refreshed/reset, but in the presence of network failures (such as network configuration information becoming stale), some fault recovering action (such as

deprecating the corresponding addresses and subsequently removing them) is triggered.

* In the context of [RFC8028], where it is clear that the use of addresses configured for a given prefix is tied to the next-hop router that advertised the prefix, the "Preferred Lifetime" of a PIO should not be larger than the "Router Lifetime" of Router Advertisement messages. Some leeway should be provided for the "Valid Lifetime" of PIOs, to cope with transient network problems. As a result, this document updates [RFC4861] such that the default Valid Lifetime (AdvValidLifetime) and the default Preferred Lifetime (AdvPreferredLifetime) of PIOs are specified as a function of the "Router Lifetime" (AdvDefaultLifetime) of Router Advertisement messages. In the absence of RAs that refresh information, addresses configured for previously-advertised prefixes become deprecated in a timelier manner, and thus Rule 3 of [RFC6724] will cause other configured addresses (if available) to be preferred.

* The expression above computes the maximum between AdvDefaultLifetime and "3 * MaxRtrAdvInterval" (the default value of AdvDefaultLifetime, as per [RFC4861]) to cope with the case where an operator might simply want to disable one local router for maintenance, without disabling the use of the corresponding prefixes on the local network (e.g., on a multi-router network). [RFC4862] implementations would otherwise deprecate the corresponding prefixes. Similarly, [RFC8028] implementations would likely behave in the same way.

4.2. Honor Small PIO Valid Lifetimes

The entire item "e)" (pp. 19-20) from Section 5.5.3 of [RFC4862] is replaced with the following text:

> e) If the advertised prefix is equal to the prefix of an address configured by stateless autoconfiguration in the list, the valid lifetime and the preferred lifetime of the address should be updated by processing the Valid Lifetime and the Preferred Lifetime (respectively) in the received advertisement.

RATIONALE:

* This change allows hosts to react to the information provided by a router that has positive knowledge that a prefix has become invalid.

* The behavior described in RFC4862 had been incorporated during the revision of the original IPv6 Stateless Address

Autoconfiguration specification ([RFC1971]). At the time, the IPNG working group decided to mitigate the attack vector represented by Prefix Information Options with very short lifetimes, on the premise these packets represented a bigger risk than other ND-based attack vectors [IPNG-minutes].

While reconsidering the trade-offs represented by such decision, we conclude that the drawbacks of mitigating the aforementioned attack vector outweigh the possible benefits.

In scenarios where RA-based attacks are of concern, proper mitigations such as RA-Guard [RFC6105] [RFC7113] or SEND [RFC3971] should be implemented.

4.3. Interface Initialization

When an interface is initialized, it is paramount that network configuration information is spread on the corresponding network (particularly in scenarios where an interface has been re-initialized, and the conveyed information has changed). Thus, this document replaces the following text from Section 6.2.4 of [RFC4861]:

> In such cases, the router MAY transmit up to MAX_INITIAL_RTR_ADVERTISEMENTS unsolicited advertisements, using the same rules as when an interface becomes an advertising interface.

with:

> In such cases, the router SHOULD transmit MAX_INITIAL_RTR_ADVERTISEMENTS unsolicited advertisements, using the same rules as when an interface becomes an advertising interface.

RATIONALE:

* Use of stale information can lead to interoperability problems. Therefore, it is important that new configuration information propagates in a timelier manner to all hosts.

NOTE:
[I-D.ietf-v6ops-cpe-slaac-renum] specifies recommendations for CPE routers to deprecate any stale network configuration information.

4.4. Conveying Information in Router Advertisement (RA) Messages

[TBD]

4.5. Recovery from Stale Configuration Information without Explicit Signaling

[TBD]

5. IANA Considerations

This document has no actions for IANA.

6. Implementation Status

[NOTE: This section is to be removed by the RFC-Editor before this document is published as an RFC.]

This section summarizes the implementation status of the updates proposed in this document. In some cases, they correspond to variants of the mitigations proposed in this document (e.g., use of reduced default lifetimes for PIOs, albeit using different values than those recommended in this document). In such cases, we believe these implementations signal the intent to deal with the problems described in [I-D.ietf-v6ops-slaac-renum] while lacking any guidance on the best possible approach to do it.

6.1. More Appropriate Lifetime Values

6.1.1. Router Configuration Variables

6.1.1.1. rad(8)

We have produced a patch for OpenBSD's rad(8) [rad] that employs the default lifetimes recommended in this document, albeit it has not yet been committed to the tree. The patch is available at: <https://www.gont.com.ar/code/fgont-patch-rad-pio-lifetimes.txt>.

6.1.1.2. radvd(8)

The radvd(8) daemon [radvd], normally employed by Linux-based router implementations, currently employs different default lifetimes than those recommended in [RFC4861]. radvd(8) employs the following default values [radvd.conf]:

- o Preferred Lifetime: 14400 seconds (4 hours)

- o Valid Lifetime: 86400 seconds (1 day)

This is not following the specific recommendation in this document, bu is already a deviation from the current standards.

6.2. Honor Small PIO Valid Lifetimes

6.2.1. Linux Kernel

A Linux kernel implementation of this document has been committed to the net-next tree. The implementation was produced in April 2020 by Fernando Gont <fgont@si6networks.com>. The corresponding patch can be found at: <https://patchwork.ozlabs.org/project/netdev/patch/20200419122457.GA971@archlinux-current.localdomain/>

6.2.2. NetworkManager

NetworkManager [NetworkManager] processes RA messages with a Valid Lifetime smaller than two hours as recommended in this document.

6.3. Conveying Information in Router Advertisement (RA) Messages

We know of no implementation that splits network configuration information into multiple RA messages.

6.4. Recovery from Stale Configuration Information without Explicit Signaling

6.4.1. dhcpcd(8)

The dhcpcd(8) daemon [dhcpcd], a user-space SLAAC implementation employed by some Linux-based and BSD-derived operating systems, will set the Preferred Lifetime of addresses corresponding to a given prefix to 0 when a single RA from the router that previously advertised the prefix fails to advertise the corresponding prefix. However, it does not affect the corresponding Valid Lifetime. Therefore, it can be considered a partial implementation of this feature.

6.5. Other mitigations implemented in products

[FRITZ] is a Customer Edge Router that tries to deprecate stale prefixes by advertising stale prefixes with a Preferred Lifetime of 0, and a Valid Lifetime of 2 hours (or less). There are two things to note with respect to this implementation:

o Rather than recording prefixes on stable storage (as recommended in [I-D.ietf-v6ops-cpe-slaac-renum]), this implementation checks the source address of IPv6 packets, and assumes that usage of any address that does not correspond to a prefix currently-advertised by the Customer Edge Router is the result of stale network configuration information. Hence, upon receipt of a packet that employs a source address that does not correspond to a currently-advertised prefix, this implementation will start advertising the corresponding prefix with small lifetimes, with the intent of deprecating it.

- Possibly as a result of item "e)" (pp. 19-20) from Section 5.5.3 of [RFC4862] (discussed in Section 4.2 of this document), upon first occurrence of a stale prefix, this implementation will employ a decreasing Valid Lifetime, starting from 2 hours (7200 seconds), as opposed to a Valid Lifetime of 0.

7. Security Considerations

The protocol update in Section 4.2 could allow an on-link attacker to perform a Denial of Service attack against local hosts, by sending a forged RA with a PIO with a Valid Lifetime of 0. Upon receipt of that packet, local hosts would invalidate the corresponding prefix, and therefore remove any addresses configured for that prefix, possibly terminating e.g. TCP connections employing such addresses. However, an attacker may achieve similar effects via a number for ND-based attack vectors, such as directing traffic to a non-existing node until ongoing TCP connections time out, or performing a ND-based man-in-the-middle (MITM) attack and subsequently forging TCP RST segments to cause on-going TCP connections to be aborted. Thus, for all practical purposes, this attack vector does not really represent a greater risk than other ND attack vectors. As noted in Section 4.2 , in scenarios where RA-based attacks are of concern, proper mitigations such as RA-Guard [RFC6105] [RFC7113] or SEND [RFC3971] should be implemented.

8. Acknowledgments

The authors would like to thank (in alphabetical order) Mikael Abrahamsson, Tore Anderson, Luis Balbinot, Brian Carpenter, Lorenzo Colitti, Owen DeLong, Gert Doering, Thomas Haller, Nick Hilliard, Bob Hinden, Philip Homburg, Lee Howard, Christian Huitema, Tatuya Jinmei, Erik Kline, Ted Lemon, Jen Linkova, Albert Manfredi, Roy Marples, Florian Obser, Jordi Palet Martinez, Michael Richardson, Hiroki Sato, Mark Smith, Hannes Frederic Sowa, Dave Thaler, Tarko Tikan, Ole Troan, Eduard Vasilenko, and Loganaden Velvindron, for providing valuable comments on earlier versions of this document.

The algorithm specified in Section 4.5 is the result of mailing-list discussions over previous versions of this document with Philip Homburg.

Fernando would like to thank Alejandro D'Egidio and Sander Steffann for a discussion of these issues, which led to the publication of [I-D.ietf-v6ops-slaac-renum], and eventually to this document.

Fernando would also like to thank Brian Carpenter who, over the years, has answered many questions and provided valuable comments that has benefited his protocol-related work.

9. References

9.1. 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>.

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

[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, <https://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, <https://www.rfc-editor.org/info/rfc4862>.

[RFC7772] Yourtchenko, A. and L. Colitti, "Reducing Energy Consumption of Router Advertisements", BCP 202, RFC 7772, DOI 10.17487/RFC7772, February 2016, <https://www.rfc-editor.org/info/rfc7772>.

[RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <https://www.rfc-editor.org/info/rfc8028>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8190] Bonica, R., Cotton, M., Haberman, B., and L. Vegoda, "Updates to the Special-Purpose IP Address Registries", BCP 153, RFC 8190, DOI 10.17487/RFC8190, June 2017, <https://www.rfc-editor.org/info/rfc8190>.

9.2. Informative References

[dhcpcd] Marples, R., "dhcpcd - a DHCP client", <https://roy.marples.name/projects/dhcpcd/>.

[FRITZ] Gont, F., "Quiz: Weird IPv6 Traffic on the Local Network (updated with solution)", SI6 Networks Blog, February 2016, <https://www.si6networks.com/2016/02/16/quiz-weird-ipv6-traffic-on-the-local-network-updated-with-solution/>.

[I-D.ietf-v6ops-cpe-slaac-renum] Gont, F., Zorz, J., Patterson, R., and B. Volz, "Improving the Reaction of Customer Edge Routers to Renumbering Events", draft-ietf-v6ops-cpe-slaac-renum-06 (work in progress), December 2020.

[I-D.ietf-v6ops-slaac-renum] Gont, F., Zorz, J., and R. Patterson, "Reaction of Stateless Address Autoconfiguration (SLAAC) to Flash-Renumbering Events", draft-ietf-v6ops-slaac-renum-05 (work in progress), November 2020.

[IPNG-minutes] IETF, "IPNG working group (ipngwg) Meeting Minutes", Proceedings of the thirty-eightt Internet Engineering Task Force , April 1997, <https://www.ietf.org/proceedings/38/97apr-final/xrtftr47.htm>.

[NetworkManager] NetworkManager, "NetworkManager web site", <https://wiki.gnome.org/Projects/NetworkManager>.

[rad] Obser, F., "OpenBSD Router Advertisement Daemon - rad(8)", <https://cvsweb.openbsd.org/src/usr.sbin/rad/>.

[radvd] Hawkins, R. and R. Johnson, "Linux IPv6 Router Advertisement Daemon (radvd)", <http://www.litech.org/radvd/>.

[radvd.conf]
Hawkins, R. and R. Johnson, "radvd.conf - configuration file of the router advertisement daemon", <https://github.com/reubenhwk/radvd/blob/master/radvd.conf.5.man>.

[RFC1971] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 1971, DOI 10.17487/RFC1971, August 1996, <https://www.rfc-editor.org/info/rfc1971>.

[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, <https://www.rfc-editor.org/info/rfc2827>.

[RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <https://www.rfc-editor.org/info/rfc3971>.

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

[RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <https://www.rfc-editor.org/info/rfc6105>.

[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, <https://www.rfc-editor.org/info/rfc6724>.

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

[slaacd] Obser, F., "OpenBSD SLAAC Daemon - slaacd(8)", <https://cvsweb.openbsd.org/src/usr.sbin/slaacd/>.

[systemd] systemd, "systemd web site", <https://systemd.io/>.

Appendix A. Analysis of Some Suggested Workarounds

[This section is to be removed before publication of this document as an RFC].

During the discussion of this document, some alternative workarounds were suggested on the 6man mailing-list. The following subsections analyze these suggested workarounds, in the hopes of avoiding rehashing the same discussions.

A.1. On a Possible Reaction to ICMPv6 Error Messages

It has been suggested that if configured addresses become stale, a CPE enforcing ingress/egress filtering (BCP38) ([RFC2827]) could send ICMPv6 Type 1 (Destination Unreachable) Code 5 (Source address failed ingress/egress policy) error messages to the sending node, and that, upon receipt of such error messages, the sending node could perform heuristics that might help to mitigate the problem discussed in this document.

The aforementioned proposal has a number of drawbacks and limitations:

- o It assumes that the CPE routers enforce ingress/egress filtering [RFC2827]. While this is desirable behaviour, it cannot be relied upon.

- o It assumes that if the CPE enforces ingress/egress filtering, the CPE will signal the packet drops to the sending node with ICMPv6 Type 1 (Destination Unreachable) Code 5 (Source address failed ingress/egress policy) error messages. While this may be desirable, [RFC2827] does not suggest signaling the packet drops with ICMPv6 error messages, let alone the use of specific error messages (such as Type 1 Code 5) as suggested.

- o ICMPv6 Type 1 Code 5 could be interpreted as the employed address being stale, but also as a selected route being inappropriate/ suboptimal. If the later, deprecating addresses or invalidating addresses upon receipt of these error messages would be inappropriate.

- o Reacting to these error messages would create a new attack vector that could be exploited from remote networks. This is of particular concern since ICMP-based attacks do not even require that the Source Address of the attack packets be spoofed [RFC5927].

A.2. On a Possible Improvement to Source Address Selection

[RFC6724] specifies source address selection (SAS) for IPv6. Conceptually, it sorts the candidate set of source addresses for a given destination, based on a number of pair-wise comparison rules that must be successively applied until there is a "winning" address.

An implementation might improve source address selection, and prefer the most-recently advertised information. In order to incorporate the "freshness" of information in source address selection, an implementation would be updated as follows:

- o The node is assumed to maintain a timer/counter that is updated at least once per second. For example, the time(2) function from unix-like systems could be employed for this purpose.

- o The local information associated with each prefix advertised via RAs on the local network is augmented with a "LastAdvertised" timestamp value. Whenever an RA with a PIO with the "A" bit set for such prefix is received, the "LastAdvertised" timestamp is updated with the current value of the timer/counter.

- o [RFC6724] is updated such that this rule is incorporated:

  Rule 7.5: Prefer fresh information If one of the two source addresses corresponds to a prefix that has been more recently advertised, say LastAdvertised(SA) > LastAdvertised(SA), then prefer that address (SA in our case).

A clear benefit of this approach is that a host will normally prefer "fresh" addresses over possibly stale addresses.

However, there are a number of drawbacks associated with this approach:

- o In scenarios where multiple prefixes are being advertised on the same LAN segment, the new SAS rule is *guaranteed* to result in non-deterministic behaviour, with hosts frequently changing the default source address. This is certainly not desirable from a troubleshooting perspective.

- o Since the rule must be incorporated before "Rule 8: Use longest matching prefix" from [RFC6724], it may lead to suboptimal paths.

- o This new rule may help to improve the selection of a source address, but it does not help with the housekeeping (garbage collection) of configured information:

* If the stale prefix is re-used in another network, nodes employing stale addresses and routes for this prefix will be unable to communicate with the new "owner" of the prefix, since the stale prefix will most likely be considered "on-link".

* Given that the currently recommended default value for the "Valid Lifetime" of PIOs is 2592000 seconds (30 days), it would take too long for hosts to remove the configured addresses and routes for the stale prefix. While the proposed update in Section 4.1 of this document would mitigate this problem, the lifetimes advertised by the local SLAAC router are not under the control of hosts.

As a result, updating IPv6 source address selection does not relieve nodes from improving their SLAAC implementations as specified in Section 4, if at all desirable. On the other hand, the algorithm specified in Section 4.5 would result in Rule 3 of [RFC6724] employing fresh addresses, without leading to non-deterministic behaviour.

Authors' Addresses

Fernando Gont
SI6 Networks
Segurola y Habana 4310, 7mo Piso
Villa Devoto, Ciudad Autonoma de Buenos Aires
Argentina

Email: fgont@si6networks.com
URI: https://www.si6networks.com

Jan Zorz
6connect

Email: jan@connect.com

Richard Patterson
Sky UK

Email: richard.patterson@sky.uk

Network Working Group T. Lemon
Internet-Draft Apple, Inc.
Intended status: Informational July 8, 2020
Expires: January 9, 2021

Self-configuring Stub Networks: Problem Statement
draft-lemon-stub-networks-ps-00

Abstract

IETF currently provides protocols for automatically connecting single hosts to existing network infrastructure. This document describes a related problem: the problem of connecting a stub network (a collection of hosts behind a router) automatically to existing network infrastructure in the same manner.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes the problem of linking stub networks to existing networks automatically, in the same way that hosts, when connected to an existing network, are able to discover network addressing parameters, information about routing, and services that are advertised on the network.

There are several use cases for stub networks. Motivating factors include:

- o Transitory connectivity: a mobile device acting as a router for a set of co-located devices could connect to a network and gain access to services for itself and for the co-located devices. Such a stub network is unlikely to have more than one stub router.

- o Incompatible media: for example, a constrained 802.15.4 network connected as a stub network to a WiFi or ethernet infrastructure network. In the case of an 802.15.4 network, it is quite possible that the devices used to link the infrastructure network to the stub network will not be conceived of by the end user as routers. Consequently, we cannot assume that these devices will be on all the time. A solution for this use case will require some sort of commissioning process for stub routers, and can't assume that any particular stub router will always be available; rather, any stub router that is available must be able to adapt to current conditions to provide reachability.

- o Convenience: end users often connect devices to each other in order to extend networks

What makes stub networks a distinct type of network is simply that a stub network never provides transit between networks to which it is connected. The term "stub" refers to the way the network is seen by the link to which it is connected: there is reachability through a stub network router to the stub network from that link, but there is no reachability to any link beyond that one.

Stub networks may be globally reachable, or may be only locally reachable. A host on a globally reachable stub network can interoperate with other hosts anywhere on the Internet. A host on a locally reachable stub network can only interoperate with hosts on the network link(s) to which it is connected.

The goal of this document is to describe the minimal set of changes or behaviors required to use existing IETF specifications to support the stub network use case. The result should be a small set of protocol enhancements (ideally no changes at all to protocols) and should be deployable on existing networks without requiring changes to those networks. Both the locally-reachable and globally-reachable use case should be able to be made to work, and ideally the globally-reachable use case should build on what is used to make the locally-reachable use case work, rather than requiring two separate solutions.

1.1. Interoperability Goals

What we mean by "interoperate" is that a host on a stub network:

- o is discoverable by applicable hosts that are not on the stub network
- o is able to acquire an IP address that can be used to communicate with applicable hosts not on the stub network
- o has reachability to the network(s) to which applicable hosts are attached
- o is reachable from the network(s) to which applicable hosts are attached

Discoverability here means "discoverable using DNS, or DNS Service Discovery". As an example, when one host connected to a specific WiFi network wishes to discover services on hosts connected to that same WiFi network, it can do so using multicast DNS (RFC6762), which is an example of DNS Service Discovery. Similarly, when a host on some other network wishes to discover the same service, it must use DNS-based DNS Service Discovery [RFC6763]. In both cases, "discoverable using DNS" means that the host has an entry in the DNS.

NOTE: it may be tempting to ask, why do we lump discoverability in with reachability and addressability, both of which are essentially Layer 3 issues? The answer is that it does us no good to automatically set up connectivity between stub network hosts and infrastructure hosts if the infrastructure hosts have no mechanism for learning about the availability of services provided by stub network hosts. For stub networks that only consume cloud services this will not be an issue, but for stub networks that provide services, e.g. the incompatible media use case mentioned earlier, discoverability is necessary in order for stub network connectivity to be useful.

Ability to acquire an IP address that can be used to communicate means that the IP address a host on the stub network acquires can be used to communicate with it by hosts on neighbor networks, for locally reachable stub networks, or by hosts on any network, for globally reachable networks. Various means of providing such addresses are discussed later.

Reachability to networks on which applicable hosts are attached means that when a host on the stub network has the IP address of an applicable host with which it intends to communicate, that host knows of a next-hop router to which it can send datagrams, so that they will ultimately reach the host with that IP address.

Reachability from networks on which applicable hosts are attached means that when such a host has a datagram destined for an IP address on the stub network, a next-hop router is known by that host which, when the datagram is sent to that router, will ultimately result in the datagram reaching the intended stub network host.

1.2. Usability Goals

In addition to the interoperability goals we've described above, the additional goal for stub networks is that they be able to be connected automatically, with no user intervention. The experience of connecting a stub network to an infrastructure should be as straightforward as connecting a new host to the same infrastructure network.

1.3. State of the Art

Currently there is one known way to accomplish what we are describing here [[Michael, does ANIMA have a second way?]]. The Homenet working group produced a protocol, HomeNet Configuration Protocol (HNCP), the purpose of which is to allow a collection of routers to self-configure. HNCP is not technically constrained to home environments; in principle, it can work in any environment.

The problem with HNCP is twofold. First, it only works if it is deployed on all routers within the network infrastructure for a site. Secondly, it attempts to do too much, and invents too much that is new. Let's look at these in order.

First, HNCP only works when deployed on all routers within the network infrastructure. To be clear, this does not mean that it is impossible to use HNCP on a network where, for instance, the edge router(s) do not support HNCP. What it does mean is that if this configuration works, the reason it works is that the network supports prefix delegation to routers inside the network. So a router doing HNCP can get a prefix using prefix delegation from, for example, an edge router, and this will work.

Unfortunately, the way that such an HNCP server should behave is not documented, and it's not actually clear how it should behave. What if the DHCP server allocates it a /64? HNCP is designed to get a larger prefix and subdivide it--there is no provision for requesting multiple delegations. So if we wanted to use HNCP to solve this problem, we would need to do additional work.

Secondly, HNCP tries to do too much, and invents too much that is new. HNCP is a complicated protocol for propagating network configuration information in a mesh. It does not assume that any network is a stub network, and because of that, using it to support stub networks is needlessly complicated.

Despite having been an IETF proposed standard since 2016, and having been worked on for quite some time before that, it is not possible to purchase a router that implements HNCP. There exists a prototype implementation in OpenWRT, but getting it to actually work is problematic, and many problems have been left unsolved, and would be quite difficult to solve with additional standards work.

We know this because several participants in the Homenet Working Group have tried to implement make it work, and yet as yet we have made no documentable progress, and indeed the Homenet Working Group appears to be on the verge of closing.

Because of the first point--the utter lack of commercial implementations of HNCP--any stub network solution that is intended to be deployed to arbitrary networks can’t rely on the availability of HNCP. This may come in the future, but is not available now, and may never be. Therefore, whatever approach is taken MAY use HNCP if available, but MUST work without HNCP. Therefore, using HNCP represents additional implementation complexity; whether this is worth doing is something that should be considered, but because using HNCP is necessarily optional, it probably makes the most sense to assume that any functionality provided by HNCP will be external to the stub network router, and that the stub network router itself need not participate in the HNCP mesh.

2. Possible Approaches

2.1. Proxy ND

2.1.1. Reachability

Proxy Neighbor Discovery provides reachability to hosts on the stub network by simply pretending that they are on the infrastructure network. This reachability can be local or global depending on what IPv6 service (if any) is available on the infrastructure link. The use of Proxy ND for providing connectivity to stub networks is described in [I-D.ietf-6lo-backbone-router].

2.1.2. Addressability

If IPv6 service is available on the infrastructure link, this service can be used to provide addressability on the stub network, and also provides addressability on the infrastructure link.

If IPv6 service is not available on the infrastructure link, addressability for proxy ND can be provided by advertising an on-link autoconfigurable prefix in a Router Advertisement offered by the stub router.

2.1.3. Discoverability

Discoverability for stub network hosts can be provided using DNS-SD service registration protocol on the stub network, in combination with an Advertising Proxy on the stub router which would advertise registered services to the infrastructure link.

Discoverability of infrastructure link hosts by stub network hosts can be provided using a DNS-SD discovery proxy and/or regular DNS. As long as the stub network requires that each stub router provide a DNS-SD Discovery Proxy and also provide name resolution, this will work even in the multiple stub router case.

2.1.4. Requirements

- The infrastructure must either provide IPv6 service, or not block the provision of IPv6 service by the stub router.
- Hosts on the infrastructure link must support IPv6 and must support IPv6 neighbor discovery.
- Every stub host must register with at least one stub router that will do proxy ND for it.
- Routers must share proxy ND information, or else each router is a single point of failure for the set of hosts that have registered with it.
- Sharing proxy ND information requires new protocol work

2.1.5. Observations

Can definitely work in specific circumstances, but probably doesn't lend itself to full automation.

2.2. Stub reachability using RA

2.2.1. Reachability

Reachability to the stub network is provided using the Route Information Option [RFC4191] in a router advertisement [RFC4861] issued by the stub router. Since the stub router does not provide IPv6 connectivity, it must not advertise itself as a default router. Each stub router can provide a default route to the stub network.

2.2.2. Addressability

Addressability on the stub network is provided using a ULA prefix generated by the stub router. Addressibility on the infrastructure link is either provided by the infrastructure, or else must be provided by the stub router.

2.2.3. Discoverability

Discoverability for this approach is the same as for the Proxy ND approach.

2.2.4. Requirements

- Infrastructure network must not block router advertisements.

- Hosts on the infrastructure network must support IPv6, must support the use of non-default routes as described in [RFC4191], and must support routing through non-default routers (routers with a router lifetime of 0).

- Stub routers must cooperate with other stub routers in announcing an on-link prefix to the stub network.

- Stub routers must cooperate with infrastructure routers in announcing an on-link prefix for the infrastructure network. Stub routers must not advertise an on-link prefix when an on-link prefix is already present.

2.2.5. Observations

This option has the advantage of relying primarily on ordinary IPv6 routing, as opposed to workarounds like proxy neighbor discovery or NAT64. The cooperation that is required between stub routers is minimal: they need simply minimize the advertising of redundant information. When redundant information is advertised, this is an aesthetic issue rather than an operational issue, and can be allowed to heal gradually.

Additionally, this option does not require any new behavior on the part of existing hosts or routers. It does assume that infrastructure hosts actually implement [RFC4191], but it is not unreasonable to expect that this either is already the case, or can easily be accomplished. It also assumes that the infrastructure does not enforce RA Guard [RFC6105]. This is compatible with the recommendations in RFC6105, which indicates that RA guard needs to be configured before it is enabled.

The approach described in this section only makes it possible for stub network hosts to interoperate with hosts on the link to which the stub router is directly attached. The "Global Reachability" approach talks about how to establish interoperability between stub network hosts and hosts on links to which the stub network is not directly attached.

2.3. Global reachability

Global reachability for stub networks requires either the use of NAT64, or else the presence of global IPv6 service on the link. As such it is more of an add-on approach than a different approach. This section talks about a specific example of global reachability: how to make global reachability work for the "Stub Reachability using RA" approach mentioned earlier.

The "global reachability" approach has applicability both in the literal sense, and also in the sense of "reachability beyond the link to which the stub router is directly attached." The behavior of the stub router is the same in both cases: it is up to the network infrastructure what prefix is delegated to the stub router, and what reachability is provided.

2.3.1. Reachability

Reachability in this case requires integration into the routing infrastructure. This is most easily accomplished by having the DHCPv6 prefix delegation server add an entry in the routing table pointing to the stub router to which the prefix has been delegated. Stub routers can also advertise reachability to the stub network using router advertisements, but these will only work on the local link.

2.3.2. Addressability

Addressability in this case for hosts on the infrastructure link is assumed to be provided by the infrastructure, since we are relying on the infrastructure to provide DHCPv6 prefix delegation.

Addressibility on the stub network is provided using the prefix acquired with prefix delegation.

2.3.3. Discoverability

Discoverability for devices on the link to which the stub network is attached can be done as described earlier under the "Proxy ND" approach.

2.3.4. Requirements

- o Infrastructure network must support prefix allocation using DHCPv6 prefix delegation.

- o Infrastructure network must install routes to prefixes provided using DHCPv6 prefix delegation.

- o In the case of multiple stub routers, stub routers must cooperate both in acquiring and renewing prefixes acquired using prefix delegation. Stub routers must communicate complete routing information to the DHCPv6 prefix delegation server so that it can install routes.

2.3.5. Observations

This approach should be a proper superset of the "Stub Reachability using RA" approach. The primary technical challenge here is specifying how multiple stub routers cooperate in doing prefix delegation.

2.4. Support for IPv4

This document generally assumes that stub networks only support IPv6. Bidirectional reachability for IPv4 can be provided using a combination of NAT44 and Port Control Protocol [RFC6887]. The use of NAT44 and PCP in this way has already been solved and need not be discussed here.

2.4.1. Reachability

Reachability is complicated for NAT64. Typical NAT64 deployments provide reachability from the stub network to the rest of the Internet, but do not provide reachability from the rest of the internet to the stub network. As with NAT44 and PCP, this type of reachability is a solved problem and need not be discussed here. To provide complete reachability to the IPv4 internet, a stub router must not only provide reachability to the cloud, but also reachability from the cloud. That additional work is discussed here.

To provide reachability from the cloud to devices on the network, devices on the network will need to obtain static mappings from the external IPv4 address and a port to the internal IPv6 address and a port. There are three ways to do this:

- o The stub host can use Port Control Protocol to register a port, and then advertise that using SRP.
- o The stub host can simply register using SRP, and then SRP can establish a port mapping.

The first option has the advantage that the stub host is in complete control over what is advertised. However, it places an additional burden on the stub host which may not be desirable: the host has to implement PCP and link the PCP port allocation to the SRP registration.

For a constrained network device, it is most likely preferable to combine the two transactions: the SRP server can receive the registration from the stub host and acquire a PCP mapping for it, and then register an AAAA and A record for the host along with an SRV record for the IPv4 and IPv6 mappings. The hostname mapping would need to be different for the A record and the AAAA record in order to avoid spurious connections to the IPv4 port on the IPv6 address and vice versa.

2.4.2. Addressability

Addressability on the stub network can be provided using a ULA prefix specific to the stub network or, if NAT64 is being used in addition to one of the other solutions discussed here, the prefix allocated on the stub network for that purpose can also be used for NAT64.

IPv4 addressability on the infrastructure network is provided by the infrastructure network. It is also possible that the infrastructure network is an IPv6 network. In that case, the NAT64 edge router may be provided by the infrastructure as well.

2.4.3. Discoverability

The discoverability described for the "ND Proxy" approach should work here as well, except for the caveat mentioned above under "reachability".

2.4.4. Requirements

- TBD

2.4.5. Observations

Support for NAT64 may be required for some deployments. NAT64 support requires either close cooperation between stub routers, or else requires that the NAT64 translation be done externally. The latter choice is likely quite a bit easier; solutions that provide load balancing and high availability are already available on the market, and hence do not require that the stub routers perform this function. This is expected to be the best approach to serve the needs of consumers of this capability.

3. Discoverability Options

We can divide the set of hosts needing to be discovered and the set of hosts needing to discover them into four categories:

- Stub network hosts (stub hosts)
- Hosts that are on the link to which the stub network is directly connected (direct hosts)
- Hosts that are on other links within the same infrastructure (infrastructure hosts)
- Hosts that are on other links not within the same infrastructure (cloud hosts)

To enable stub hosts to discover direct hosts, a Discovery Proxy [RFC8766] can be used. This must be resident on any stub network router that is seen by the stub host as a resolver.

To enable stub hosts to discover infrastructure hosts using DNS-SD [RFC6763], the infrastructure must provide support for RFC6763 service discover using DNS.

To enable stub hosts to discover infrastructure hosts and cloud hosts using DNS, DNS resolution must be provided by the stub router, and the infrastructure must additionally provide the stub router with the ability to resolve names.

To enable direct hosts to discover stub hosts, stub routers must implement a DNS-SD Advertising Proxy. Stub hosts must register with the advertising proxy using SRP.

To enable infrastructure hosts to discover stub hosts, stub routers must provide authoritative DNS service for the stub network link so that it can be integrated into the infrastructure DNS-SD service. To do this automatically will require additional protocol work.

To enable cloud hosts to discover stub hosts, stub hosts would need to register with the DNS, and the infrastructure would need to make those registrations available globally, perhaps with whitelisting. This is probably not a very widely applicable use case, and we do not consider specifying how this works to be part of the work of this document.

4. Multiple Egress, Multiple Link

In the case of a stub network that has multiple stub routers, it is possible that, either when the stub network is initially set up, or subsequently, one or more stub routers might be connected to a different infrastructure link than one or more other stub routers. There are two viable approaches to this problem:

- declare it out of scope and have the stub routers prevent such configurations

- make sure that stub routers attached to each infrastructure link provide complete service on that link

Explain further.

5. Management Considerations

TBD

6. Privacy Considerations

In the locally reachable case, privacy is protected in the sense that names published locally are only visible to devices connected locally. This may be insufficient privacy in some cases.

In the globally reachable case, discoverability has privacy implications. Unfiltered automatic discoverability is probably not a good idea in the globally reachable case. If automatic discoverability is provided, some filtering mechanism would need to be specified.

7. Security Considerations

TBD

8. IANA considerations

No new actions are required by IANA for this document.

9. Informative References

[I-D.ietf-6lo-backbone-router]
Thubert, P., Perkins, C., and E. Levy-Abegnoli, "IPv6 Backbone Router", draft-ietf-6lo-backbone-router-20 (work in progress), March 2020.

[I-D.sctl-service-registration]
Cheshire, S. and T. Lemon, "Service Registration Protocol for DNS-Based Service Discovery", draft-sctl-service-registration-02 (work in progress), July 2018.

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

[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, <https://www.rfc-editor.org/info/rfc4861>.

[RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <https://www.rfc-editor.org/info/rfc6105>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <https://www.rfc-editor.org/info/rfc6887>.

[RFC8766] Cheshire, S., "Discovery Proxy for Multicast DNS-Based Service Discovery", RFC 8766, DOI 10.17487/RFC8766, June 2020, <https://www.rfc-editor.org/info/rfc8766>.

Author's Address

Ted Lemon
Apple, Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: elemon@apple.com

China Mobile
July 13, 2020

IPv6 hosts detection
draft-li-6man-6hosts-detection-00

Abstract

The management of hosts and risks is important for enterprises that have large scale IP space. For IPv4, it won’t take too long even to scan the entire Internet address space. For IPv6, further consideration is needed. A narrow range of IPv6 address is preferred for scanning. And in order to shorten the time for IPv6 scanning, a very specific IPv6 address list is highly needed.

This document proposes a solution to solve the problem. At first, append the information of the collection point address to the Router Advertisement packet sent by the router, and announce this address information to all nodes in the subnet. Then, each host node report its own IPv6 address information to the designated collection point by using Echo Reply message. After that, the corresponding collection point device should save these information. In this way, online IPv6 address information in the current network can be quickly collected on the collection point device.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119][RFC8174].

Status of This Memo

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 14, 2021.

Copyright Notice

Table of Contents

1. Introduction

IP scanning is widely used in cybersecurity to find out online hosts and detect risks. Detection for online IPv6 hosts quickly and effectively is much more complicated than IPv4. Complications arise both from IPv6's address assignment features, e.g., stateless address autoconfiguration (SLAAC, [RFC4862]), and from the large scale IP space. The management of IPv6 hosts is difficult. This document proposes a solution to shorten the time to scan IPv6.

2. Terminology

This document uses the terminology defined in [[RFC4443]] and [[RFC4861]].

Host - any node that is not a router.

Router - a node that forwards IP packets not explicitly addressed to itself.

Node - a device that implements IP.

In addition, there is a new term that is defined below.

Collection Point - a device with a global IPv6 address that can store information.

3. Message Formats

3.1. Router Advertisement Option Formats

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Valid Lifetime                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                   Collection Point Address                    +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

Type 39. It is 8-bit identifier of the Collection Point option type.

Length 3.

Reserved This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Valid Lifetime 32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that the address is valid for the purpose of on-link determination. A value of all one bits (0xffffffff) represents infinity.

Collection Point Address
A 128-bit IPv6 address of the Collection Point.

3.2. Echo Reply Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Identifier          |       Squence Number          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Data ...
+-+-+-+-+-+-+-+-+
```

IPv6 Fields:

Destination Address
A 128-bit IPv6 address of the Collection Point.

ICMPv6 Fields:

Type	129
Code	0
Identifier	0xffff
Sequence Number	1
Data	Special tag content is set. The default value is COLLECTION ONLY

4. Online Address Collection

4.1. Router Specification

4.1.1. Router Configuration Variables

AdvCollectionPoint

A global IPv6 address to be placed in Collection Point Information options in Router Advertisement messages sent from the interface.

Default: all Collection Point that the router advertises via routing protocols as being on-link for the interface from which the advertisement is sent.

The link-local address SHOULD NOT be included in the list of advertised address.

Each Collection Point has an associated:

AdvValidLifetime

The value to be placed in the Valid Lifetime in the Collection Point Information option, in seconds. The designated value of all 1's (0xffffffff) represents infinity.

Implementations MAY allow AdvValidLifetime to be specified in two ways:

- a time that decrements in real time, that is, one that will result in a Lifetime of zero at the specified time in the future, or

- a fixed time that stays the same in consecutive advertisements.

Default: 2592000 seconds (30 days), fixed(i.e., stays the same in consecutive advertisements).

4.1.2. Router Advertisement Message Content

The details of the technical part of Router Advertisement of the router are the same as the relevant provisions in RFC 4861. When there is a Collection Point Address in the router, the router should carry the content information of Collection Point Address in the option of the Router Advertisement Message, with the message format given in Section 3.1.

4.2. Host Specification

4.2.1. Processing Received Router Advertisements and Sending Echo Reply

When a host receives the Router Advertisement sent by the router, and finds that there is the information of Collection Point Address in the Router Advertisement, the host delays a random time, and then an Echo Reply should be sent to Collection Point.

The specific information of the Echo Reply packet is as follows. The destination address is the Collection Point Address, and the source address is the global unicast address of the host.

The Data in the Echo Reply packet contains special tag content, which is COLLECTION ONLY defined in Section 3.2.

The frequency of the Echo Reply packet sent by the host is the same as the frequency of receiving valid Router Advertisement packets which contains the information of Collection Point Address

When the host interface is used as a router in any other network, the device needs to transfer the information of Collection Point Address received by the host to its AdvCollectionPoint parameter as a router node

4.3. Collection Point Specification

When the Collection Point receives an Echo Reply packet while it doesn’t actively send any Echo Request packet, it should extract the source address of this Echo Reply packet, which should be a global unicast address. And save the source address by attaching the current system timestamp.

5. Security Considerations

Because RAs are required in all IPv6 configuration scenarios, on IPv6-only networks, RAs must already be secured -- e.g., by deploying an RA-Guard [[RFC6105]]. Providing all configuration in RAs reduces the attack surface to be targeted by malicious attackers trying to provide hosts with invalid configuration, as compared to distributing the configuration through multiple different mechanisms that need to be secured independently.

Connectivity to destinations reachable over IPv6 would not be impacted just by providing a host with an incorrect Collection Point address; however, if attackers are capable of sending rogue RAs, they can perform denial-of-service or man-in-the-middle attacks, as described in [[RFC6104]].

6. IANA Considerations

IANA has assigned a new IPv6 Neighbor Discovery Option type for the Collection Point option defined in this document in the "IPv6 Neighbor Discovery Option Formats" registry [IANA].

Description	Type
Collection Point option	39

Table 1: New IANA Registry Assignment

7. References

7.1. 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>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://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, <https://www.rfc-editor.org/info/rfc4861>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[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>.

[RFC6104] Chown, T. and S. Venaas, "Rogue IPv6 Router Advertisement Problem Statement", RFC 6104, DOI 10.17487/RFC6104, February 2011, <https://www.rfc-editor.org/info/rfc6104>.

[RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <https://www.rfc-editor.org/info/rfc6105>.

Authors' Addresses

Jiang Li
China Mobile
Beijing 100053
China

Email: lijiang@chinamobile.com

Jun Fu
China Mobile
Beijing 100053
China

Email: fujun@chinamobile.com

Xiaoxiao Li
China Mobile
Beijing 100053
China

Email: lixiaoxiao@chinamobile.com

Yexia Cheng
China Mobile
Beijing 100053
China

Email: chengyexia@chinamobile.com

Network Working Group F. Templin, Ed.
Internet-Draft The Boeing Company
Updates: rfc1191, rfc4443, rfc8201 (if A. Whyman
approved) MWA Ltd c/o Inmarsat Global Ltd
Intended status: Standards Track January 1, 2021
Expires: July 5, 2021

Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces
draft-templin-6man-omni-interface-68

Abstract

Mobile nodes (e.g., aircraft of various configurations, terrestrial vehicles, seagoing vessels, enterprise wireless devices, etc.) communicate with networked correspondents over multiple access network data links and configure mobile routers to connect end user networks. A multilink interface specification is therefore needed for coordination with the network-based mobility service. This document specifies the transmission of IP packets over Overlay Multilink Network (OMNI) Interfaces.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Mobile Nodes (MNs) (e.g., aircraft of various configurations, terrestrial vehicles, seagoing vessels, enterprise wireless devices, pedestrians with cellphones, etc.) often have multiple interface connections to wireless and/or wired-link data links used for communicating with networked correspondents. These data links may have diverse performance, cost and availability properties that can change dynamically according to mobility patterns, flight phases, proximity to infrastructure, etc. MNs coordinate their data links in a discipline known as "multilink", in which a single virtual interface is configured over the node's underlying interface connections to the data links.

The MN configures a virtual interface (termed the "Overlay Multilink Network (OMNI) interface") as a thin layer over the underlying interfaces. The OMNI interface is therefore the only interface abstraction exposed to the IP layer and behaves according to the Non-Broadcast, Multiple Access (NBMA) interface principle, while underlying interfaces appear as link layer communication channels in the architecture. The OMNI interface connects to a virtual overlay service known as the "OMNI link". The OMNI link spans one or more Internetworks that may include private-use infrastructures and/or the global public Internet itself.

Each MN receives a Mobile Network Prefix (MNP) for numbering downstream-attached End User Networks (EUNs) independently of the access network data links selected for data transport. The MN performs router discovery over the OMNI interface (i.e., similar to IPv6 customer edge routers [RFC7084]) and acts as a mobile router on behalf of its EUNs. The router discovery process is iterated over each of the OMNI interface's underlying interfaces in order to register per-link parameters (see Section 13).

The OMNI interface provides a multilink nexus for exchanging inbound and outbound traffic via the correct underlying interface(s). The IP layer sees the OMNI interface as a point of connection to the OMNI link. Each OMNI link has one or more associated Mobility Service

Prefixes (MSPs), which are typically IP Global Unicast Address (GUA) prefixes from which OMNI link MNPs are derived. If there are multiple OMNI links, the IPv6 layer will see multiple OMNI interfaces.

MNs may connect to multiple distinct OMNI links by configuring multiple OMNI interfaces, e.g., omni0, omni1, omni2, etc. Each OMNI interface is configured over a set of underlying interfaces and provides a nexus for Safety-Based Multilink (SBM) operation. Each OMNI SBM "domain" configures a common ULA prefix [ULA]::/48, and each OMNI link within the domain configures a unique 16-bit Subnet ID '*' to construct the sub-prefix [ULA*]::/64 (see: Section 8). The IP layer applies SBM routing to select an OMNI interface, which then applies Performance-Based Multilink (PBM) to select the correct underlying interface. Applications can apply Segment Routing [RFC8402] to select independent SBM topologies for fault tolerance.

The OMNI interface interacts with a network-based Mobility Service (MS) through IPv6 Neighbor Discovery (ND) control message exchanges [RFC4861]. The MS provides Mobility Service Endpoints (MSEs) that track MN movements and represent their MNPs in a global routing or mapping system.

Many OMNI use cases are currently under active consideration. In particular, the International Civil Aviation Organization (ICAO) Working Group-I Mobility Subgroup is developing a future Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS) and has issued a liaison statement requesting IETF adoption [ATN] in support of ICAO Document 9896 [ATN-IPS]. The IETF IP Wireless Access in Vehicular Environments (ipwave) working group has further included problem statement and use case analysis for OMNI in a document now in AD evaluation for RFC publication [I-D.ietf-ipwave-vehicular-networking]. Still other communities of interest include AEEC, RTCA Special Committee 228 (SC-228) and NASA programs that examine commercial aviation, Urban Air Mobility (UAM) and Unmanned Air Systems (UAS). Pedestrians with handheld devices represent another large class of potential OMNI users.

This document specifies the transmission of IP packets and MN/MS control messages over OMNI interfaces. The OMNI interface supports either IP protocol version (i.e., IPv4 [RFC0791] or IPv6 [RFC8200]) as the network layer in the data plane, while using IPv6 ND messaging as the control plane independently of the data plane IP protocol(s). The OMNI Adaptation Layer (OAL) which operates as a mid-layer between L3 and L2 is based on IP-in-IPv6 encapsulation per [RFC2473] as discussed in the following sections.

2. Terminology

The terminology in the normative references applies; especially, the terms "link" and "interface" are the same as defined in the IPv6 [RFC8200] and IPv6 Neighbor Discovery (ND) [RFC4861] specifications. Additionally, this document assumes the following IPv6 ND message types: Router Solicitation (RS), Router Advertisement (RA), Neighbor Solicitation (NS), Neighbor Advertisement (NA) and Redirect.

The Protocol Constants defined in Section 10 of [RFC4861] are used in their same format and meaning in this document. The terms "All-Routers multicast", "All-Nodes multicast" and "Subnet-Router anycast" are the same as defined in [RFC4291] (with Link-Local scope assumed).

The term "IP" is used to refer collectively to either Internet Protocol version (i.e., IPv4 [RFC0791] or IPv6 [RFC8200]) when a specification at the layer in question applies equally to either version.

The following terms are defined within the scope of this document:

Mobile Node (MN)
: an end system with a mobile router having multiple distinct upstream data link connections that are grouped together in one or more logical units. The MN's data link connection parameters can change over time due to, e.g., node mobility, link quality, etc. The MN further connects a downstream-attached End User Network (EUN). The term MN used here is distinct from uses in other documents, and does not imply a particular mobility protocol.

End User Network (EUN)
: a simple or complex downstream-attached mobile network that travels with the MN as a single logical unit. The IP addresses assigned to EUN devices remain stable even if the MN's upstream data link connections change.

Mobility Service (MS)
: a mobile routing service that tracks MN movements and ensures that MNs remain continuously reachable even across mobility events. Specific MS details are out of scope for this document.

Mobility Service Endpoint (MSE)
: an entity in the MS (either singular or aggregate) that coordinates the mobility events of one or more MN.

Mobility Service Prefix (MSP)
: an aggregated IP Global Unicast Address (GUA) prefix (e.g., 2001:db8::/32, 192.0.2.0/24, etc.) assigned to the OMNI link and

from which more-specific Mobile Network Prefixes (MNPs) are delegated. OMNI link administrators typically obtain MSPs from an Internet address registry, however private-use prefixes can alternatively be used subject to certain limitations (see: Section 9). OMNI links that connect to the global Internet advertise their MSPs to their interdomain routing peers.

Mobile Network Prefix (MNP)
: a longer IP prefix delegated from an MSP (e.g., 2001:db8:1000:2000::/56, 192.0.2.8/30, etc.) and assigned to a MN. MNs sub-delegate the MNP to devices located in EUNs.

Access Network (ANET)
: a data link service network (e.g., an aviation radio access network, satellite service provider network, cellular operator network, wifi network, etc.) that connects MNs. Physical and/or data link level security is assumed, and sometimes referred to as "protected spectrum". Private enterprise networks and ground domain aviation service networks may provide multiple secured IP hops between the MN’s point of connection and the nearest Access Router.

Access Router (AR)
: a router in the ANET for connecting MNs to correspondents in outside Internetworks. The AR may be located on the same physical link as the MN, or may be located multiple IP hops away. In the latter case, the MN uses encapsulation to communicate with the AR as though it were on the same physical link.

ANET interface
: a MN’s attachment to a link in an ANET.

Internetwork (INET)
: a connected network region with a coherent IP addressing plan that provides transit forwarding services between ANETs and nodes that connect directly to the open INET via unprotected media. No physical and/or data link level security is assumed, therefore security must be applied by upper layers. The global public Internet itself is an example.

INET interface
: a node’s attachment to a link in an INET.

*NET
: a "wildcard" term used when a given specification applies equally to both ANET and INET cases.

OMNI link

a Non-Broadcast, Multiple Access (NBMA) virtual overlay configured over one or more INETs and their connected ANETs. An OMNI link can comprise multiple INET segments joined by bridges the same as for any link; the addressing plans in each segment may be mutually exclusive and managed by different administrative entities.

OMNI interface
a node's attachment to an OMNI link, and configured over one or more underlying *NET interfaces. If there are multiple OMNI links in an OMNI domain, a separate OMNI interface is configured for each link.

OMNI Adaptation Layer (OAL)
an OMNI interface process whereby packets admitted into the interface are wrapped in a mid-layer IPv6 header and fragmented/ reassembled if necessary to support the OMNI link Maximum Transmission Unit (MTU). The OAL is also responsible for generating MTU-related control messages as necessary, and for providing addressing context for spanning multiple segments of a bridged OMNI link.

OMNI Option
an IPv6 Neighbor Discovery option providing multilink parameters for the OMNI interface as specified in Section 10.

Mobile Network Prefix Link Local Address (MNP-LLA)
an IPv6 Link Local Address that embeds the most significant 64 bits of an MNP in the lower 64 bits of fe80::/64, as specified in Section 7.

Mobile Network Prefix Unique Local Address (MNP-ULA)
an IPv6 Unique-Local Address derived from an MNP-LLA.

Administrative Link Local Address (ADM-LLA)
an IPv6 Link Local Address that embeds a 32-bit administratively-assigned identification value in the lower 32 bits of fe80::/96, as specified in Section 7.

Administrative Unique Local Address (ADM-ULA)
an IPv6 Unique-Local Address derived from an ADM-LLA.

Multilink
an OMNI interface's manner of managing diverse underlying interface connections to data links as a single logical unit. The OMNI interface provides a single unified interface to upper layers, while underlying interface selections are performed on a per-packet basis considering factors such as DSCP, flow label, application policy, signal quality, cost, etc. Multilinking

decisions are coordinated in both the outbound (i.e. MN to correspondent) and inbound (i.e., correspondent to MN) directions.

L2
The second layer in the OSI network model. Also known as "layer-2", "link-layer", "sub-IP layer", "data link layer", etc.

L3
The third layer in the OSI network model. Also known as "layer-3", "network-layer", "IP layer", etc.

underlying interface
a *NET interface over which an OMNI interface is configured. The OMNI interface is seen as a L3 interface by the IP layer, and each underlying interface is seen as a L2 interface by the OMNI interface. The underlying interface either connects directly to the physical communications media or coordinates with another node where the physical media is hosted.

Mobility Service Identification (MSID)
Each MSE and AR is assigned a unique 32-bit Identification (MSID) (see: Section 7). IDs are assigned according to MS-specific guidelines (e.g., see: [I-D.templin-intarea-6706bis]).

Safety-Based Multilink (SBM)
A means for ensuring fault tolerance through redundancy by connecting multiple affiliated OMNI interfaces to independent routing topologies (i.e., multiple independent OMNI links).

Performance Based Multilink (PBM)
A means for selecting underlying interface(s) for packet transmission and reception within a single OMNI interface.

OMNI Domain
The set of all SBM/PBM OMNI links that collectively provides services for a common set of MSPs. Each OMNI domain consists of a set of affiliated OMNI links that all configure the same ::/48 ULA prefix with a unique 16-bit Subnet ID as discussed in Section 8.

3. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

OMNI links maintain a constant value "MAX_MSID" selected to provide MNs with an acceptable level of MSE redundancy while minimizing control message amplification. It is RECOMMENDED that MAX_MSID be set to the default value 5; if a different value is chosen, it should be set uniformly by all nodes on the OMNI link.

An implementation is not required to internally use the architectural constructs described here so long as its external behavior is consistent with that described in this document.

4. Overlay Multilink Network (OMNI) Interface Model

An OMNI interface is a MN virtual interface configured over one or more underlying interfaces, which may be physical (e.g., an aeronautical radio link) or virtual (e.g., an Internet or higher-layer "tunnel"). The MN receives a MNP from the MS, and coordinates with the MS through IPv6 ND message exchanges. The MN uses the MNP to construct a unique Link-Local Address (MNP-LLA) through the algorithmic derivation specified in Section 7 and assigns the LLA to the OMNI interface.

The OMNI interface architectural layering model is the same as in [RFC5558][RFC7847], and augmented as shown in Figure 1. The IP layer therefore sees the OMNI interface as a single L3 interface with multiple underlying interfaces that appear as L2 communication channels in the architecture.

```
                                  +----------------------------+
                                  |    Upper Layer Protocol    |
           Session-to-IP    +---->|                            |
           Address Binding  |     +----------------------------+
                            +---->|           IP (L3)          |
           IP Address       +---->|                            |
           Binding          |     +----------------------------+
                            +---->|       OMNI Interface       |
           Logical-to-      +---->|           (LLA)            |
           Physical         |     +----------------------------+
           Interface        +---->|  L2  |  L2  |       |  L2  |
           Binding                |(IF#1)|(IF#2)| ..... |(IF#n)|
                                  +------+------+       +------+
                                  |  L1  |  L1  |       |  L1  |
                                  |      |      |       |      |
                                  +------+------+       +------+
```

Figure 1: OMNI Interface Architectural Layering Model

Each underlying interface provides an L2/L1 abstraction according to one of the following models:

- o INET interfaces connect to an INET either natively or through one or several IPv4 Network Address Translators (NATs). Native INET interfaces have global IP addresses that are reachable from any INET correspondent. NATed INET interfaces typically have private IP addresses and connect to a private network behind one or more NATs that provide INET access.

- o ANET interfaces connect to a protected ANET that is separated from the open INET by an AR acting as a proxy. The ANET interface may be either on the same L2 link segment as the AR, or separated from the AR by multiple IP hops.

- o VPNed interfaces use security encapsulation over a *NET to a Virtual Private Network (VPN) gateway. Other than the link-layer encapsulation format, VPNed interfaces behave the same as for Direct interfaces.

- o Direct (aka "point-to-point") interfaces connect directly to a peer without crossing any *NET paths. An example is a line-of-sight link between a remote pilot and an unmanned aircraft.

The OMNI virtual interface model gives rise to a number of opportunities:

- o since MNP-LLAs are uniquely derived from an MNP, no Duplicate Address Detection (DAD) or Multicast Listener Discovery (MLD) messaging is necessary.

- o since Temporary LLAs are statistically unique, they can be used without DAD for short-term purposes, e.g. until an MNP-LLA is obtained.

- o *NET interfaces on the same L2 link segment as an AR do not require any L3 addresses (i.e., not even link-local) in environments where communications are coordinated entirely over the OMNI interface. (An alternative would be to also assign the same LLA to all *NET interfaces.)

- o as underlying interface properties change (e.g., link quality, cost, availability, etc.), any active interface can be used to update the profiles of multiple additional interfaces in a single message. This allows for timely adaptation and service continuity under dynamically changing conditions.

- o coordinating underlying interfaces in this way allows them to be represented in a unified MS profile with provisions for mobility and multilink operations.

- o exposing a single virtual interface abstraction to the IPv6 layer allows for multilink operation (including QoS based link selection, packet replication, load balancing, etc.) at L2 while still permitting L3 traffic shaping based on, e.g., DSCP, flow label, etc.

- o the OMNI interface allows inter-INET traversal when nodes located in different INETs need to communicate with one another. This mode of operation would not be possible via direct communications over the underlying interfaces themselves.

- o the OMNI Adaptation Layer (OAL) within the OMNI interface supports lossless and adaptive path MTU mitigations not available for communications directly over the underlying interfaces themselves.

- o L3 sees the OMNI interface as a point of connection to the OMNI link; if there are multiple OMNI links (i.e., multiple MS's), L3 will see multiple OMNI interfaces.

- o Multiple independent OMNI interfaces can be used for increased fault tolerance through Safety-Based Multilink (SBM), with Performance-Based Multilink (PBM) applied within each interface.

Other opportunities are discussed in [RFC7847]. Note that even when the OMNI virtual interface is present, applications can still access underlying interfaces either through the network protocol stack using an Internet socket or directly using a raw socket. This allows for intra-network (or point-to-point) communications without invoking the OMNI interface and/or OAL. For example, when an IPv6 OMNI interface is configured over an underlying IPv4 interface, applications can still invoke IPv4 intra-network communications as long as the communicating endpoints are not subject to mobility dynamics. However, the opportunities discussed above are not available when the architectural layering is bypassed in this way.

Figure 2 depicts the architectural model for a MN with an attached EUN connecting to the MS via multiple independent *NETs. When an underlying interface becomes active, the MN's OMNI interface sends IPv6 ND messages without encapsulation if the first-hop Access Router (AR) is on the same underlying link; otherwise, the interface uses IP-in-IP encapsulation. The IPv6 ND messages traverse the ground domain *NETs until they reach an AR (AR#1, AR#2, ..., AR#n), which then coordinates with an INET Mobility Service Endpoint (MSE#1, MSE#2, ..., MSE#m) and returns an IPv6 ND message response to the MN. The Hop Limit in IPv6 ND messages is not decremented due to encapsulation; hence, the OMNI interface appears to be attached to an ordinary link.

```
                           +--------------+        (:::)-.
                           |      MN      |<-->.-(::EUN:::)
                           +--------------+      `-(::::)-'
                           |OMNI interface|
                           +----+----+----+
                  +--------|IF#1|IF#2|IF#n|------ +
                 /         +----+----+----+        \
                /                 |                 \
               /                  |                  \
              v                   v                   v
           (:::)-.              (:::)-.              (:::)-.
      .-(::*NET:::)        .-(::*NET:::)        .-(::*NET:::)
        `-(::::)-'           `-(::::)-'           `-(::::)-'
          +----+               +----+               +----+
    ...   |AR#1|  ..........   |AR#2|   .........   |AR#n|  ...
   .      +-|--+               +-|--+               +-|--+     .
   .        |                    |                    |
   .        v                    v                    v        .
   .             <-----  INET Encapsulation ----->             .
   .                                                           .
   .      +-----+               (:::)-.                        .
   .      |MSE#2|           .-(:::::::)           +-----+      .
   .      +-----+       .-(:::   INET  :::)-.     |MSE#m|      .
   .                  (:::::    Routing  ::::)    +-----+      .
   .                     `-(::: System :::)-'                  .
   .  +-----+                `-(::::::-'                       .
   .  |MSE#1|          +-----+               +-----+           .
   .  +-----+          |MSE#3|               |MSE#4|           .
   .                   +-----+               +-----+           .
   .                                                           .
   .                                                           .
   .       <----- Worldwide Connected Internetwork ---->       .
    ...........................................................
```

Figure 2: MN/MS Coordination via Multiple *NETs

After the initial IPv6 ND message exchange, the MN (and/or any nodes on its attached EUNs) can send and receive IP data packets over the OMNI interface. OMNI interface multilink services will forward the packets via ARs in the correct underlying *NETs. The AR encapsulates the packets according to the capabilities provided by the MS and forwards them to the next hop within the worldwide connected Internetwork via optimal routes.

OMNI links span one or more underlying Internetwork via the OMNI Adaptation Layer (OAL) which is based on a mid-layer overlay encapsulation using [RFC2473]. Each OMNI link corresponds to a different overlay (differentiated by an address codepoint) which may be carried over a completely separate underlying topology. Each MN can facilitate SBM by connecting to multiple OMNI links using a distinct OMNI interface for each link.

Note: OMNI interface underlying interfaces often connect directly to physical media on the local platform (e.g., a laptop computer with WiFi, etc.), but in some configurations the physical media may be hosted on a separate Local Area Network (LAN) node. In that case, the OMNI interface can establish a Layer-2 VLAN or a point-to-point tunnel (at a layer below the underlying interface) to the node hosting the physical media. The OMNI interface may also apply encapsulation at a layer above the underlying interface such that packets would appear "double-encapsulated" on the LAN; the node hosting the physical media in turn removes the LAN encapsulation prior to transmission or inserts it following reception. Finally, the underlying interface must monitor the node hosting the physical media (e.g., through periodic keepalives) so that it can convey up/down/status information to the OMNI interface.

5. The OMNI Adaptation Layer (OAL)

The OMNI interface observes the link nature of tunnels, including the Maximum Transmission Unit (MTU), Maximum Reassembly Unit (MRU) and the role of fragmentation and reassembly [I-D.ietf-intarea-tunnels]. The OMNI interface is configured over one or more underlying interfaces that may have diverse MTUs. OMNI interfaces accommodate MTU diversity through the use of the OMNI Adaptation Layer (OAL) as discussed in this section.

IPv6 underlying interfaces are REQUIRED to configure a minimum MTU of 1280 bytes and a minimum MRU of 1500 bytes [RFC8200]. Therefore, the minimum IPv6 path MTU is 1280 bytes since routers on the path are not permitted to perform network fragmentation even though the destination is required to reassemble more. The network therefore MUST forward packets of at least 1280 bytes without generating an IPv6 Path MTU Discovery (PMTUD) Packet Too Big (PTB) message [RFC8201]. (Note: the source can apply "source fragmentation" for locally-generated IPv6 packets up to 1500 bytes and larger still if it if has a way to determine that the destination configures a larger MRU, but this does not affect the minimum IPv6 path MTU.)

IPv4 underlying interfaces are REQUIRED to configure a minimum MTU of 68 bytes [RFC0791] and a minimum MRU of 576 bytes [RFC0791][RFC1122]. Therefore, when the Don't Fragment (DF) bit in the IPv4 header is set to 0 the minimum IPv4 path MTU is 576 bytes since routers on the path support network fragmentation and the destination is required to reassemble at least that much. The "Don't Fragment" (DF) bit in the IPv4 encapsulation headers of packets sent over IPv4 underlying interfaces therefore MUST be set to 0. (Note: even if the encapsulation source has a way to determine that the encapsulation destination configures an MRU larger than 576 bytes, it should not assume a larger minimum IPv4 path MTU without careful consideration of the issues discussed in Section 5.1.)

In network paths where IPv6/IPv4 protocol translation or IPv6-in-IPv4 encapsulation may be prevalent, it may be prudent for the OAL to always assume the IPv4 minimum path MTU (i.e., 576 bytes) regardless of the underlying interface IP protocol version. Always assuming the IPv4 minimum path MTU even for IPv6 underlying interfaces may produce more fragments and additional header overhead, but will always interoperate and never run the risk of presenting an IPv4 interface with a packet that exceeds its MRU.

The OMNI interface configures both an MTU and MRU of 9180 bytes [RFC2492]; the size is therefore not a reflection of the underlying interface MTUs, but rather determines the largest packet the OMNI interface can forward or reassemble. The OMNI interface uses the OMNI Adaptation Layer (OAL) to admit packets from the network layer that are no larger than the OMNI interface MTU while generating ICMPv4 Fragmentation Needed [RFC1191] or ICMPv6 Path MTU Discovery (PMTUD) Packet Too Big (PTB) [RFC8201] messages as necessary. This document refers to both of these ICMPv4/ICMPv6 message types simply as "PTBs", and introduces a distinction between PTB "hard" and "soft" errors as discussed below.

For IPv4 packets with DF=0, the network layer performs IPv4 fragmentation if necessary then admits the packets/fragments into the OMNI interface; these fragments will be reassembled by the final destination. For IPv4 packets with DF=1 and IPv6 packets, the network layer admits the packet if it is no larger than the OMNI interface MTU; otherwise, it drops the packet and returns a PTB hard error message to the source.

For each admitted IP packet/fragment, the OMNI interface internally employs the OAL when necessary by encapsulating the inner IP packet/ fragment in a mid-layer IPv6 header per [RFC2473] before adding any outer IP encapsulations. (The OAL does not decrement the inner IP Hop Limit/TTL during encapsulation since the insertion occurs at a layer below IP forwarding.) The OAL then calculates the 32-bit CRC over the entire mid-layer packet and writes the value in a trailing 4-octet field at the end of the packet. Next, the OAL fragments this mid-layer IPv6 packet, forwards the fragments (using *NET encapsulation if necessary), and returns an internally-generated PTB soft error message (subject to rate limiting) if it deems the packet too large according to factors such as link performance characteristics, reassembly congestion, etc. This ensures that the path MTU is adaptive and reflects the current path used for a given data flow.

The OAL operates with respect to both the minimum IPv6 and IPv4 path MTUs as follows:

o When an OMNI interface sends a packet toward a final destination via an ANET peer, it sends without OAL encapsulation if the packet (including any outer-layer ANET encapsulations) is no larger than the underlying interface MTU for on-link ANET peers or the minimum ANET path MTU for peers separated by multiple IP hops. Otherwise, the OAL inserts an IPv6 header per [RFC2473] with source address set to the node's own Unique-Local Address (ULA) (see: Section 8) and destination set to either the Administrative ULA (ADM-ULA) of the ANET peer or the Mobile Network Prefix ULA (MNP-ULA) corresponding to the final destination (see below). The OAL then calculates and appends the trailing 32-bit CRC, then uses IPv6 fragmentation to break the packet into a minimum number of non-overlapping fragments where the size of each non-final fragment (including both the OMNI and any outer-layer ANET encapsulations) is determined by the underlying interface MTU for on-link ANET peers or the minimum ANET path MTU for peers separated by multiple IP hops. The OAL then encapsulates the fragments in any ANET headers and sends them to the ANET peer, which either reassembles before forwarding if the OAL destination is its own ADM-ULA or forwards the fragments toward the final destination without first reassembling otherwise.

o When an OMNI interface sends a packet toward a final destination via an INET interface, it sends packets (including any outer-layer INET encapsulations) no larger than the minimum INET path MTU without OAL encapsulation if the destination is reached via an INET address within the same OMNI link segment. Otherwise, the OAL inserts an IPv6 header per [RFC2473] with source address set to the node's ULA, destination set to the ULA of the next hop OMNI node toward the final destination and (if necessary) with an OMNI Routing Header (ORH) (see: [I-D.templin-intarea-6706bis]) with final segment addressing information. The OAL then calculates and appends the trailing 32-bit CRC, then uses IPv6 fragmentation to break the packet into a minimum number of non-overlapping fragments where the size of each non-final fragment (including both the OMNI and outer-layer INET encapsulations) is determined by the minimum INET path MTU. The OAL then encapsulates the fragments in any INET headers and sends them to the OMNI link neighbor, which reassembles before forwarding toward the final destination.

In light of the above considerations, the OAL should assume a minimum path MTU of 576 bytes for the purpose of generating OAL fragments. Each OAL fragment will undergo *NET encapsulation including either a 20 byte IPv4 or 40 byte IPv6 header plus an 8 byte UDP header, leaving a minimum of 528 bytes for each fragment. Each OAL fragment must accommodate 48 bytes for the OAL IPv6 header plus fragment header, with the remaining space available for a portion of the inner IP packet while reserving 40 additional bytes in case a maximum-length ORH is added due to re-encapsulation. OAL fragmentation algorithms must therefore produce non-final fragments that are 488 bytes in length (i.e., not including the ORH length), while the final fragment may be smaller. OAL fragmentation algorithms may produce larger non-final fragments if better path MTU information is available, but must not produce smaller non-final fragments.

Note that when two ANET peers share a common physical or virtual link with a larger MTU such as 1500 bytes or larger, OAL fragmentation may use this larger MTU size as long as the receiving ANET peer reassembles (and possibly also refragments) before forwarding. This is important for accommodating links where performance is highly dependent on maximum use of the available link MTU, e.g. for wireless aviation data links. Additionally, in order to set the correct context for reassembly, the OMNI interface that inserts the OAL header MUST also be the one that inserts the IPv6 fragment header Identification value. While not strictly required, sending all fragments of the same fragmented OAL packet consecutively over the same underlying interface with minimal inter-fragment delay may increase the likelihood of successful reassembly.

Ordinary PTB messages with ICMPv4 header "unused" field or ICMPv6 header Code field value 0 are hard errors that always indicate that a packet has been dropped due to a real MTU restriction. However, the OAL can also forward large packets via encapsulation and fragmentation while at the same time returning PTB soft error messages (subject to rate limiting) indicating that a forwarded packet was uncomfortably large. The OMNI interface can therefore continuously forward large packets without loss while returning PTB soft error messages recommending a smaller size. Original sources that receive the soft errors in turn reduce the size of the packets they send, i.e., the same as for hard errors.

The OAL sets the ICMPv4 header "unused" field or ICMPv6 header Code field to the value 1 in PTB soft error messages. The OAL sets the PTB destination address to the source address of the original packet, and sets the source address to the MNP Subnet Router Anycast address of the MN (i.e., whether the MN was the source or target of the original packet). The OAL then sets the MTU field to a value no smaller than 576 for ICMPv4 or 1280 for ICMPv6, and returns the PTB soft error to the original source.

When the original source receives the PTB, it reduces its path MTU estimate the same as for hard errors but does not regard the message as a loss indication. (If the original source does not recognize the soft error code, it regards the PTB the same as a hard error but should heed the retransmission advice given in [RFC8201] suggesting retransmission based on normal packetization layer retransmission timers.) This document therefore updates [RFC1191][RFC4443] and [RFC8201]. Furthermore, implementations of [RFC4821] must be aware that PTB hard or soft errors may arrive at any time even if after a successful MTU probe (this is the same consideration as for an ordinary path fluctuation following a successful probe).

In summary, the OAL supports continuous transmission and reception of packets of various sizes in the face of dynamically changing network conditions. Moreover, since PTB soft errors do not indicate loss, original sources that receive soft errors can quickly scan for path MTU increases without waiting for the minimum 10 minutes specified for loss-oriented PTB hard errors [RFC1191][RFC8201]. The OAL therefore provides a lossless and adaptive service that accommodates MTU diversity especially well-suited for dynamic multilink environments.

Note: An OMNI interface that reassembles OAL fragments may experience congestion-oriented loss in its reassembly cache and can optionally send PTB soft errors to the original source and/or ICMP "Time Exceeded" messages to the source of the OAL fragments. In environments where the messages may contribute to unacceptable additional congestion, however, the OMNI interface can simply regard the loss as an ordinary unreported congestion event for which the original source will eventually compensate.

Note: When the network layer forwards an IPv4 packet/fragment with DF=0 into the OMNI interface, the interface can optionally perform (further) IPv4 fragmentation before invoking the OAL so that the fragments will be reassembled by the final destination. When the network layer performs IPv6 fragmentation for locally-generated IPv6 packets, the OMNI interface typically invokes the OAL without first applying (further) IPv6 fragmentation; the network layer should therefore fragment to the minimum IPv6 path MTU (or smaller still) to push the reassembly burden to the final destination and avoid receiving PTB soft errors from the OMNI interface. Aside from these non-normative guidelines, the manner in which any IP fragmentation is invoked prior to OAL encapsulation/fragmentation is an implementation matter.

Note: Inclusion of the 32-bit CRC prior to fragmentation assumes that the receiving OAL will discard any packets with incorrect CRC values following reassembly. The 32-bit CRC is sufficient to detect reassembly misassociations for packet sizes up to the OMNI interface MTU 9180 but may not be sufficient to detect errors for larger sizes [CRC].

Note: Some underlying interface types (e.g., VPNs) may already provide their own robust fragmentation and reassembly services even without OAL encapsulation. In those cases, the OAL can invoke the inherent underlying interface schemes instead while employing PTB soft errors in the same fashion as described above. Other underlying interface properties such as header/message compression can also be harnessed in a similar fashion.

Note: Applications can dynamically tune the size of the packets they to send to produce the best possible throughput and latency, with the understanding that these parameters may change over time due to factors such as congestion, mobility, network path changes, etc. The receipt or absence of soft errors should be seen as hints of when increasing or decreasing packet sizes may be beneficial.

5.1. Fragmentation Security Implications

As discussed in Section 3.7 of [RFC8900], there are four basic threats concerning IPv6 fragmentation; each of which is addressed by effective mitigations as follows:

1. Overlapping fragment attacks - reassembly of overlapping fragments is forbidden by [RFC8200]; therefore, this threat does not apply to the OAL.

2. Resource exhaustion attacks - this threat is mitigated by providing a sufficiently large OAL reassembly cache and instituting "fast discard" of incomplete reassemblies that may be part of a buffer exhaustion attack. The reassembly cache should be sufficiently large so that a sustained attack does not cause excessive loss of good reassemblies but not so large that (timer-based) data structure management becomes computationally expensive. The cache should also be indexed based on the arrival underlying interface such that congestion experienced over a first underlying interface does not cause discard of incomplete reassemblies for uncongested underlying interfaces.

3. Attacks based on predictable fragment identification values - this threat is mitigated by selecting a suitably random ID value per [RFC7739].

4. Evasion of Network Intrusion Detection Systems (NIDS) - this threat is mitigated by disallowing "tiny fragments" per the OAL fragmentation procedures specified above.

Additionally, IPv4 fragmentation includes a 16-bit Identification (IP ID) field with only 65535 unique values such that at high data rates the field could wrap and apply to new packets while the fragments of old packets using the same ID are still alive in the network [RFC4963]. However, since the largest OAL fragment that will be sent via an IPv4 *NET path is 576 bytes any IPv4 fragmentation would occur only on links with an IPv4 MTU smaller than this size, and [RFC3819] recommendations suggest that such links will have low data rates. Since IPv6 provides a 32-bit Identification value, IP ID wraparound at high data rates is not a concern for IPv6 fragmentation.

6. Frame Format

The OMNI interface transmits IPv6 packets according to the native frame format of each underlying interface. For example, for Ethernet-compatible interfaces the frame format is specified in [RFC2464], for aeronautical radio interfaces the frame format is specified in standards such as ICAO Doc 9776 (VDL Mode 2 Technical Manual), for tunnels over IPv6 the frame format is specified in [RFC2473], etc.

7. Link-Local Addresses (LLAs)

OMNI nodes are assigned OMNI interface IPv6 Link-Local Addresses (LLAs) through pre-service administrative actions. "MNP-LLAs" embed the MNP assigned to the mobile node, while "ADM-LLAs" include an administratively-unique ID that is guaranteed to be unique on the link. LLAs are configured as follows:

- IPv6 MNP-LLAs encode the most-significant 64 bits of a MNP within the least-significant 64 bits of the IPv6 link-local prefix fe80::/64, i.e., in the LLA "interface identifier" portion. The prefix length for the LLA is determined by adding 64 to the MNP prefix length. For example, for the MNP 2001:db8:1000:2000::/56 the corresponding MNP-LLA is fe80::2001:db8:1000:2000/120.

- IPv4-compatible MNP-LLAs are constructed as fe80::ffff:[IPv4], i.e., the interface identifier consists of 16 '0' bits, followed by 16 '1' bits, followed by a 32bit IPv4 address/prefix. The prefix length for the LLA is determined by adding 96 to the MNP prefix length. For example, the IPv4-Compatible MN OMNI LLA for 192.0.2.0/24 is fe80::ffff:192.0.2.0/120 (also written as fe80::ffff:c000:0200/120).

o ADM-LLAs are assigned to ARs and MSEs and MUST be managed for uniqueness. The lower 32 bits of the LLA includes a unique integer "MSID" value between 0x00000001 and 0xfeffffff, e.g., as in fe80::1, fe80::2, fe80::3, etc., fe80::feffffff. The ADM-LLA prefix length is determined by adding 96 to the MSID prefix length. For example, if the prefix length for MSID 0x10012001 is 16 then the ADM-LLA prefix length is set to 112 and the LLA is written as fe80::1001:2001/112. The "zero" address for each ADM-LLA prefix is the Subnet-Router anycast address for that prefix [RFC4291]; for example, the Subnet-Router anycast address for fe80::1001:2001/112 is simply fe80::1001:2000. The MSID range 0xff000000 through 0xffffffff is reserved for future use.

o Temporary LLAs are constructed per [I-D.ietf-6man-rfc4941bis] and used by MNs for the short-term purpose of procuring an actual MNP-LLA upon startup or (re)connecting to the network. MNs may use Temporary LLAs as the IPv6 source address of an RS message in order to request a MNP-LLA from the MS.

Since the prefix 0000::/8 is "Reserved by the IETF" [RFC4291], no MNPs can be allocated from that block ensuring that there is no possibility for overlap between the various LLA constructs discussed above.

Since MNP-LLAs are based on the distribution of administratively assured unique MNPs, and since ADM-LLAs are guaranteed unique through administrative assignment, OMNI interfaces set the autoconfiguration variable DupAddrDetectTransmits to 0 [RFC4862].

Temporary LLAs employ optimistic DAD principles [RFC4429] since they are probabilistically unique and their use is short-duration in nature.

Note: If future protocol extensions relax the 64-bit boundary in IPv6 addressing, the additional prefix bits of an MNP could be encoded in bits 16 through 63 of the MNP-LLA. (The most-significant 64 bits would therefore still be in bits 64-127, and the remaining bits would appear in bits 16 through 48.) However, the analysis provided in [RFC7421] suggests that the 64-bit boundary will remain in the IPv6 architecture for the forseeable future.

Note: Even though this document honors the 64-bit boundary in IPv6 addressing per [RFC7421], it suggests prefix lengths longer than /64 for routing purposes. This effectively extends IPv6 routing determination into the interface identifier portion of the IPv6 address, but it does not redefine the 64-bit boundary.

8. Unique-Local Addresses (ULAs)

OMNI domains use IPv6 Unique-Local Addresses (ULAs) as the source and destination addresses in OAL IPv6 encapsulation headers. ULAs are only routable within the scope of a an OMNI domain, and are derived from the IPv6 Unique Local Address prefix fc00::/7 followed by the L bit set to 1 (i.e., as fd00::/8) followed by a 40-bit pseudo-random Global ID to produce the prefix [ULA]::/48, which is then followed by a 16-bit Subnet ID then finally followed by a 64 bit Interface ID as specified in Section 3 of [RFC4193]. The statistic uniqueness of the 40-bit pseudo-random Global ID allows different OMNI domains to be joined together in the future without requiring OMNI link renumbering.

Each OMNI link instance is identified by a value between 0x0000 and 0xfeff in bits 48-63 of [ULA]::/48 (the values 0xff00 through 0xffff are reserved for future use). For example, OMNI ULAs associated with instance 0 are configured from the prefix [ULA]:0000::/64, instance 1 from [ULA]:0001::/64, instance 2 from [ULA]:0002::/64, etc. ULAs and their associated prefix lengths are configured in correspondence with LLAs through stateless prefix translation where "MNP-ULAs" are assigned in correspondence to MNP-LLAs and "ADM-ULAs" are assigned in correspondence to ADM-LLAs. For example, for OMNI link instance [ULA]:1010::/64:

- o the MNP-ULA corresponding to the MNP-LLA fe80::2001:db8:1:2 with a 56-bit MNP length is derived by copying the lower 64 bits of the LLA into the lower 64 bits of the ULA as [ULA]:1010:2001:db8:1:2/120 (where, the ULA prefix length becomes 64 plus the IPv6 MNP length).

- o the MNP-ULA corresponding to fe80::ffff:192.0.2.0 with a 28-bit MNP length is derived by simply writing the LLA interface ID into the lower 64 bits as [ULA]:1010:0:ffff:192.0.2.0/124 (where, the ULA prefix length is 64 plus 32 plus the IPv4 MNP length).

- o the ADM-ULA corresponding to fe80::1000/112 is simply [ULA]:1010::1000/112.

- o the ADM-ULA corresponding to fe80::/128 is simply [ULA]:1010::/128.

- o the Temporary ULA corresponding to a Temporary LLA is simply [ULA]:1010:[64-bit Temporary Interface ID]/128.

- o etc.

Each OMNI interface assigns the Anycast ADM-ULA specific to the OMNI link instance. For example, the OMNI interface connected to instance 3 assigns the Anycast address [ULA]:0003::/128. Routers that configure OMNI interfaces advertise the OMNI service prefix (e.g., [ULA]:0003::/64) into the local routing system so that applications can direct traffic according to SBM requirements.

The ULA presents an IPv6 address format that is routable within the OMNI routing system and can be used to convey link-scoped IPv6 ND messages across multiple hops using IPv6 encapsulation [RFC2473]. The OMNI link extends across one or more underling Internetworks to include all ARs and MSEs. All MNs are also considered to be connected to the OMNI link, however OAL encapsulation is omitted whenever possible to conserve bandwidth (see: Section 12).

Each OMNI link can be subdivided into "segments" that often correspond to different administrative domains or physical partitions. OMNI nodes can use IPv6 Segment Routing [RFC8402] when necessary to support efficient packet forwarding to destinations located in other OMNI link segments. A full discussion of Segment Routing over the OMNI link appears in [I-D.templin-intarea-6706bis].

Note: IPv6 ULAs taken from the prefix fc00::/7 followed by the L bit set to 0 (i.e., as fc00::/8) are never used for OMNI OAL addressing, however the range could be used for MSP and MNP addressing under certain limiting conditions (see: Section 9).

9. Global Unicast Addresses (GUAs)

OMNI domains use IP Global Unicast Address (GUA) prefixes [RFC4291] as Mobility Service Prefixes (MSPs) from which Mobile Network Prefixes (MNP) are delegated to Mobile Nodes (MNs).

For IPv6, GUA prefixes are assigned by IANA [IPV6-GUA] and/or an associated regional assigned numbers authority such that the OMNI domain can be interconnected to the global IPv6 Internet without causing inconsistencies in the routing system. An OMNI domain could instead use ULAs with the 'L' bit set to 0 (i.e., from the prefix fc00::/8)[RFC4193], however this would require IPv6 NAT if the domain were ever connected to the global IPv6 Internet.

For IPv4, GUA prefixes are assigned by IANA [IPV4-GUA] and/or an associated regional assigned numbers authority such that the OMNI domain can be interconnected to the global IPv4 Internet without causing routing inconsistencies. An OMNI domain could instead use private IPv4 prefixes (e.g., 10.0.0.0/8, etc.) [RFC3330], however this would require IPv4 NAT if the domain were ever connected to the global IPv4 Internet.

10. Address Mapping - Unicast

OMNI interfaces maintain a neighbor cache for tracking per-neighbor state and use the link-local address format specified in Section 7. OMNI interface IPv6 Neighbor Discovery (ND) [RFC4861] messages sent over physical underlying interfaces without encapsulation observe the native underlying interface Source/Target Link-Layer Address Option (S/TLLAO) format (e.g., for Ethernet the S/TLLAO is specified in [RFC2464]). OMNI interface IPv6 ND messages sent over underlying interfaces via encapsulation do not include S/TLLAOs which were intended for encoding physical L2 media address formats and not encapsulation IP addresses. Furthermore, S/TLLAOs are not intended for encoding additional interface attributes needed for multilink coordination. Hence, this document does not define an S/TLLAO format but instead defines a new option type termed the "OMNI option" designed for these purposes.

MNs such as aircraft typically have many wireless data link types (e.g. satellite-based, cellular, terrestrial, air-to-air directional, etc.) with diverse performance, cost and availability properties. The OMNI interface would therefore appear to have multiple L2 connections, and may include information for multiple underlying interfaces in a single IPv6 ND message exchange. OMNI interfaces use an IPv6 ND option called the OMNI option formatted as shown in Figure 3:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |T|  Preflen    |  S/T-ifIndex  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                          Sub-Options                          ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: OMNI Option Format

In this format:

- o Type is set to TBD. If multiple OMNI option instances appear in the same IPv6 ND message, the first instance is processed and all other instances are ignored.
- o Length is set to the number of 8 octet blocks in the option.
- o T is a 1-bit flag set to 1 for Temporary LLAs (otherwise, set to 0) and Preflen is a 7 bit field that determines the length of

prefix associated with an LLA. Values 1 through 127 specify a prefix length, while the value 0 indicates "unspecified". For IPv6 ND messages sent from a MN to the MS, T and Preflen apply to the IPv6 source LLA and provide the length that the MN is requesting or asserting to the MS. For IPv6 ND messages sent from the MS to the MN, T and Preflen apply to the IPv6 destination LLA and indicate the length that the MS is granting to the MN. For IPv6 ND messages sent between MS endpoints, T is set to 0 and Preflen provides the length associated with the source/target MN that is subject of the ND message.

- o S/T-ifIndex corresponds to the ifIndex value for source or target underlying interface used to convey this IPv6 ND message. OMNI interfaces MUST number each distinct underlying interface with an ifIndex value between '1' and '255' that represents a MN-specific 8-bit mapping for the actual ifIndex value assigned by network management [RFC2863] (the ifIndex value '0' is reserved for use by the MS). For RS and NS messages, S/T-ifIndex corresponds to the source underlying interface the message originated from. For RA and NA messages, S/T-ifIndex corresponds to the target underlying interface that the message is destined to. (For NS messages used for Neighbor Unreachability Detection (NUD), S/T-ifIndex instead identifies the neighbor's underlying interface to be used as the target interface to return the NA.)

- o Sub-Options is a Variable-length field, of length such that the complete OMNI Option is an integer multiple of 8 octets long. Contains one or more Sub-Options, as described in Section 10.1.

10.1. Sub-Options

The OMNI option includes zero or more Sub-Options. Each consecutive Sub-Option is concatenated immediately after its predecessor. All Sub-Options except Pad1 (see below) are in type-length-value (TLV) encoded in the following format:

```
 0                   1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|   Sub-Type    |  Sub-length   | Sub-Option Data ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 4: Sub-Option Format

- o Sub-Type is a 1-octet field that encodes the Sub-Option type. Sub-Options defined in this document are:

```
        Option Name                 Sub-Type
        Pad1                            0
        PadN                            1
        Interface Attributes            2
        Traffic Selector                3
        MS-Register                     4
        MS-Release                      5
        Network Access Identifier       6
        Geo Coordinates                 7
        DHCP Unique Identifier (DUID)   8
        DHCPv6 Message                  9
```

Figure 5

Sub-Types 253 and 254 are reserved for experimentation, as recommended in [RFC3692].

- o Sub-Length is a 1-octet field that encodes the length of the Sub-Option Data (i.e., ranging from 0 to 255 octets).

- o Sub-Option Data is a block of data with format determined by Sub-Type.

During processing, unrecognized Sub-Options are ignored and the next Sub-Option processed until the end of the OMNI option is reached.

The following Sub-Option types and formats are defined in this document:

10.1.1. Pad1

```
         0
         0 1 2 3 4 5 6 7
        +-+-+-+-+-+-+-+-+
        |   Sub-Type=0  |
        +-+-+-+-+-+-+-+-+
```

Figure 6: Pad1

- o Sub-Type is set to 0. If multiple instances appear in the same OMNI option all are processed.

- o No Sub-Length or Sub-Option Data follows (i.e., the "Sub-Option" consists of a single zero octet).

10.1.2. PadN

```
 0                   1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|  Sub-Type=1   | Sub-length=N  | N padding octets ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 7: PadN

- o Sub-Type is set to 1. If multiple instances appear in the same OMNI option all are processed.
- o Sub-Length is set to N (from 0 to 255) being the number of padding octets that follow.
- o Sub-Option Data consists of N zero-valued octets.

10.1.3. Interface Attributes

The Interface Attributes sub-option provides L2 forwarding information for the multilink conceptual sending algorithm discussed in Section 12. The L2 information is used for selecting among potentially multiple candidate underlying interfaces that can be used to forward packets to the neighbor based on factors such as DSCP preferences and link quality. Interface Attributes further include link-layer address information to be used for either OAL encapsulation or direct UDP/IP encapsulation (when OAL encapsulation can be avoided). The Interface Attributes format and contents are given in Figure 8 below:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Sub-Type=2   | Sub-length=N  |    ifIndex    |    ifType     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Provider ID   | Link  |R| API |    SRT    | FMT |  LHS (0 - 7)  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |               LHS (bits 8 - 31)                |               ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                ~
   ~                                                                ~
   ~                   Link Layer Address (L2ADDR)                  ~
   ~                                                                ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Bitmap(0)=0xff|P00|P01|P02|P03|P04|P05|P06|P07|P08|P09|P10|P11|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |P12|P13|P14|P15|P16|P17|P18|P19|P20|P21|P22|P23|P24|P25|P26|P27|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |P28|P29|P30|P31| Bitmap(1)=0xff|P32|P33|P34|P35|P36| ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 8: Interface Attributes

- o Sub-Type is set to 2. If multiple instances with different ifIndex values appear in the same OMNI option all are processed; if multiple instances with the same ifIndex value appear, the first is processed and all others are ignored.

- o Sub-Length is set to N (from 4 to 255) that encodes the number of Sub-Option Data octets that follow.

- o Sub-Option Data contains an "Interface Attribute" option encoded as follows (note that the first four octets must be present):

  * ifIndex is set to an 8-bit integer value corresponding to a specific underlying interface the same as specified above for the OMNI option header S/T-ifIndex. An OMNI option may include multiple Interface Attributes Sub-Options, with each distinct ifIndex value pertaining to a different underlying interface. The OMNI option will often include an Interface Attributes Sub-Option with the same ifIndex value that appears in the S/T-ifIndex. In that case, the actual encapsulation address of the received IPv6 ND message should be compared with the L2ADDR encoded in the Sub-Option (see below); if the addresses are different (or, if L2ADDR is absent) the presence of a NAT is assumed.

  * ifType is set to an 8-bit integer value corresponding to the underlying interface identified by ifIndex. The value

represents an OMNI interface-specific 8-bit mapping for the actual IANA ifType value registered in the 'IANAifType-MIB' registry [http://www.iana.org].

* Provider ID is set to an OMNI interface-specific 8-bit ID value for the network service provider associated with this ifIndex.

* Link encodes a 4-bit link metric. The value '0' means the link is DOWN, and the remaining values mean the link is UP with metric ranging from '1' ("lowest") to '15' ("highest").

* R is reserved for future use.

* API - a 3-bit "Address/Preferences/Indexed" code that determines the contents of the remainder of the sub-option as follows:

  + When the most significant bit (i.e., "Address") is set to 1, the SRT, FMT, LHS and L2ADDR fields are included immediately following the API code; else, they are omitted.

  + When the next most significant bit (i.e., "Preferences") is set to 1, a preferences block is included next; else, it is omitted. (Note that if "Address" is set the preferences block immediately follows L2ADDR; else, it immediately follows the API code.)

  + When a preferences block is present and the least significant bit (i.e., "Indexed") is set to 0, the block is encoded in "Simplex" form as shown in Figure 8; else it is encoded in "Indexed" form as discussed below.

* When API indicates that an "Address" is included, the following fields appear in consecutive order (else, they are omitted):

  + SRT - a 5-bit Segment Routing Topology prefix length value that (when added to 96) determines the prefix length to apply to the ULA formed from concatenating [ULA*]::/96 with the 32 bit LHS MSID value that follows. For example, the value 16 corresponds to the prefix length 112.

  + FMT - a 3-bit "Framework/Mode/Type" code corresponding to the included Link Layer Address as follows:

    - When the most significant bit (i.e., "Framework") is set to 0, L2ADDR is the INET encapsulation address of a Proxy/Server; otherwise, it is the address for the Source/Target itself

- When the next most significant bit (i.e., "Mode") is set to 0, the Source/Target L2ADDR is on the open INET; otherwise, it is (likely) located behind one or more NATs.

- When the least significant bit (i.e., "Type") is set to 0, L2ADDR includes a UDP Port Number followed by an IPv4 address; else, a UDP Port Number followed by an IPv6 address.

+ LHS - the 32 bit MSID of the Last Hop Server/Proxy on the path to the target. When SRT and LHS are both set to 0, the LHS is considered unspecified in this IPv6 ND message. When SRT is set to 0 and LHS is non-zero, the prefix length is set to 128. SRT and LHS together provide guidance to the OMNI interface forwarding algorithm. Specifically, if SRT/LHS is located in the local OMNI link segment then the OMNI interface can encapsulate according to FMT/L2ADDR (following any necessary NAT traversal messaging); else, it must forward according to the OMNI link spanning tree. See [I-D.templin-intarea-6706bis] for further discussion.

+ Link Layer Address (L2ADDR) - Formatted according to FMT, and identifies the link-layer address (i.e., the encapsulation address) of the source/target. The UDP Port Number appears in the first two octets and the IP address appears in the next 4 octets for IPv4 or 16 octets for IPv6. The Port Number and IP address are recorded in ones-compliment "obfuscated" form per [RFC4380]. The OMNI interface forwarding algorithm uses FMT/L2ADDR to determine the encapsulation address for forwarding when SRT/LHS is located in the local OMNI link segment. Note that if the target is behind a NAT, L2ADDR will contain the mapped INET address stored in the NAT; otherwise, L2ADDR will contain the native INET information of the target itself.

* When API indicates that "Preferences" are included, a preferences block appears as the remainder of the Sub-Option as a series of Bitmaps and P[*] values. In "Simplex" form, the index for each singleton Bitmap octet is inferred from its sequential position (i.e., 0, 1, 2, ...) as shown in Figure 8. In "Indexed" form, each Bitmap is preceded by an Index octet that encodes a value "i" = (0 - 255) as the index for its companion Bitmap as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|   Index=i     |   Bitmap(i)   |P[*] values ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 9

* The preferences consist of a first (simplex/indexed) Bitmap (i.e., "Bitmap(i)") followed by 0-8 single-octet blocks of 2-bit P[*] values, followed by a second Bitmap (i), followed by 0-8 blocks of P[*] values, etc. Reading from bit 0 to bit 7, the bits of each Bitmap(i) that are set to '1'' indicate the P[*] blocks from the range P[(i*32)] through P[(i*32) + 31] that follow; if any Bitmap(i) bits are '0', then the corresponding P[*] block is instead omitted. For example, if Bitmap(0) contains 0xff then the block with P[00]-P[03], followed by the block with P[04]-P[07], etc., and ending with the block with P[28]-P[31] are included (as shown in Figure 8). The next Bitmap(i) is then consulted with its bits indicating which P[*] blocks follow, etc. out to the end of the Sub-Option.

* Each 2-bit P[*] field is set to the value '0' ("disabled"), '1' ("low"), '2' ("medium") or '3' ("high") to indicate a QoS preference for underlying interface selection purposes. Not all P[*] values need to be included in the OMNI option of each IPv6 ND message received. Any P[*] values represented in an earlier OMNI option but omitted in the current OMNI option remain unchanged. Any P[*] values not yet represented in any OMNI option default to "medium".

* The first 16 P[*] blocks correspond to the 64 Differentiated Service Code Point (DSCP) values P[00] - P[63] [RFC2474]. Any additional P[*] blocks that follow correspond to "pseudo-DSCP" traffic classifier values P[64], P[65], P[66], etc. See Appendix A for further discussion and examples.

10.1.4. Traffic Selector

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |   Sub-Type=3  | Sub-length=N  |    ifIndex    |               ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               ~
       ~                                                               ~
       ~                 RFC 6088 Format Traffic Selector              ~
       ~                                                               ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: Traffic Selector

- o Sub-Type is set to 3. If multiple instances appear in the same OMNI option all are processed, i.e., even if the same ifIndex value appears multiple times.

- o Sub-Length is set to N (the number of Sub-Option Data octets that follow).

- o Sub-Option Data contains a 1-octet ifIndex encoded exactly as specified in Section 10.1.3, followed by an N-1 octet traffic selector formatted per [RFC6088] beginning with the "TS Format" field. The largest traffic selector for a given ifIndex is therefore 254 octets.

10.1.5. MS-Register

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |   Sub-Type=4  | Sub-length=4n |     MSID[1] (bits 0 - 15)     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     MSID [1] (bits 16 - 32)   |     MSID[2] (bits 0 - 15)     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     MSID [2] (bits 16 - 32)   |     MSID[3] (bits 0 - 15)     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           ...        ...        ...        ...       ...        ...
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     MSID [n] (bits 16 - 32)   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: MS-Register Sub-option

- o Sub-Type is set to 4. If multiple instances appear in the same OMNI option all are processed. Only the first MAX_MSID values processed (whether in a single instance or multiple) are retained and all other MSIDs are ignored.

o Sub-Length is set to 4n.

o A list of n 4-octet MSIDs is included in the following 4n octets. The Anycast MSID value '0' in an RS message MS-Register sub-option requests the recipient to return the MSID of a nearby MSE in a corresponding RA response.

10.1.6. MS-Release

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Type=5  | Sub-length=4n |      MSID[1] (bits 0 - 15)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     MSID [1] (bits 16 - 32)   |      MSID[2] (bits 0 - 15)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     MSID [2] (bits 16 - 32)   |      MSID[3] (bits 0 - 15)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ...        ...        ...        ...       ...        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     MSID [n] (bits 16 - 32)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: MS-Release Sub-option

o Sub-Type is set to 5. If multiple instances appear in the same IPv6 OMNI option all are processed. Only the first MAX_MSID values processed (whether in a single instance or multiple) are retained and all other MSIDs are ignored.

o Sub-Length is set to 4n.

o A list of n 4 octet MSIDs is included in the following 4n octets. The Anycast MSID value '0' is ignored in MS-Release sub-options, i.e., only non-zero values are processed.

10.1.7. Network Access Identifier (NAI)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Type=6  | Sub-length=N  |Network Access Identifier (NAI)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ...
```

Figure 13: Network Access Identifier (NAI) Sub-option

o Sub-Type is set to 6. If multiple instances appear in the same OMNI option the first is processed and all others are ignored.

o Sub-Length is set to N (i.e., the length of the encoded Network Access Identifier (NAI)).

o An NAI up to 255 octets in length is coded per [RFC7542].

10.1.8. Geo Coordinates

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Type=7  | Sub-length=N  |      Geo Coordinates
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ...
```

Figure 14: Geo Coordinates Sub-option

o Sub-Type is set to 7. If multiple instances appear in the same OMNI option the first is processed and all others are ignored.

o Sub-Length is set to N (i.e., the length of the encoded Geo Coordinates).

o A set of Geo Coordinates up to 255 octets in length (format TBD). Includes Latitude/Longitude at a minimum; may also include additional attributes such as altitude, heading, speed, etc.).

10.1.9. DHCP Unique Identifier (DUID)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Type=8  | Sub-length=N  |           DUID-Type           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.             type-specific DUID body (variable length)         .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: DHCP Unique Identifier (DUID) Sub-option

o Sub-Type is set to 8. If multiple instances appear in the same OMNI option the first is processed and all others are ignored.

o Sub-Length is set to N (i.e., the length of the option beginning with the DUID-Type and continuing to the end of the type-specific body).

o DUID-Type is a two-octet field coded in network byte order that determines the format and contents of the type-specific body according to Section 11 of [RFC8415]. DUID-Type 4 in particular corresponds to the Universally Unique Identifier (UUID) [RFC6355] which will occur in common operational practice.

- A type-specific DUID body up to 253 octets in length follows, formatted according to DUID-type. For example, for type 4 the body consists of a 128-bit UUID selected according to [RFC6355].

10.1.10. DHCPv6 Message

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Type=9  | Sub-length=N  |    msg-type   |  id (octet 0) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   transaction-id (octets 1-2) |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
.                        DHCPv6 options                         .
.                 (variable number and length)                  .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: DHCPv6 Message Sub-option

- Sub-Type is set to 9. If multiple instances appear in the same OMNI option the first is processed and all others are ignored.
- Sub-Length is set to N (i.e., the length of the DHCPv6 message beginning with 'msg-type' and continuing to the end of the DHCPv6 options). The length of the entire DHCPv6 message is therefore restricted to 255 octets.
- 'msg-type' and 'transaction-id' are coded according to Section 8 of [RFC8415].
- A set of DHCPv6 options coded according to Section 21 of [RFC8415] follows.

11. Address Mapping - Multicast

The multicast address mapping of the native underlying interface applies. The mobile router on board the MN also serves as an IGMP/MLD Proxy for its EUNs and/or hosted applications per [RFC4605] while using the L2 address of the AR as the L2 address for all multicast packets.

The MN uses Multicast Listener Discovery (MLDv2) [RFC3810] to coordinate with the AR, and *NET L2 elements use MLD snooping [RFC4541].

12. Multilink Conceptual Sending Algorithm

The MN's IPv6 layer selects the outbound OMNI interface according to SBM considerations when forwarding data packets from local or EUN applications to external correspondents. Each OMNI interface maintains a neighbor cache the same as for any IPv6 interface, but with additional state for multilink coordination. Each OMNI interface maintains default routes via ARs discovered as discussed in Section 13, and may configure more-specific routes discovered through means outside the scope of this specification.

After a packet enters the OMNI interface, one or more outbound underlying interfaces are selected based on PBM traffic attributes, and one or more neighbor underlying interfaces are selected based on the receipt of Interface Attributes sub-options in IPv6 ND messages (see: Figure 8). Underlying interface selection for the nodes own local interfaces are based on attributes such as DSCP, application port number, cost, performance, message size, etc. OMNI interface multilink selections could also be configured to perform replication across multiple underlying interfaces for increased reliability at the expense of packet duplication. The set of all Interface Attributes received in IPv6 ND messages determine the multilink forwarding profile for selecting the neighbor's underlying interfaces.

When the OMNI interface sends a packet over a selected outbound underlying interface, the OAL includes or omits a mid-layer encapsulation header as necessary as discussed in Section 5 and as determined by the L2 address information received in Interface Attributes. The OAL also performs encapsulation when the nearest AR is located multiple hops away as discussed in Section 13.1.

OMNI interface multilink service designers MUST observe the BCP guidance in Section 15 [RFC3819] in terms of implications for reordering when packets from the same flow may be spread across multiple underlying interfaces having diverse properties.

12.1. Multiple OMNI Interfaces

MNs may connect to multiple independent OMNI links concurrently in support of SBM. Each OMNI interface is distinguished by its Anycast ULA (e.g., [ULA]:0002::, [ULA]:1000::, [ULA]:7345::, etc.). The MN configures a separate OMNI interface for each link so that multiple interfaces (e.g., omni0, omni1, omni2, etc.) are exposed to the IPv6 layer. A different Anycast ULA is assigned to each interface, and the MN injects the service prefixes for the OMNI link instances into the EUN routing system.

Applications in EUNs can use Segment Routing to select the desired OMNI interface based on SBM considerations. The Anycast ULA is written into the IPv6 destination address, and the actual destination (along with any additional intermediate hops) is written into the Segment Routing Header. Standard IP routing directs the packets to the MN's mobile router entity, and the Anycast ULA identifies the OMNI interface to be used for transmission to the next hop. When the MN receives the message, it replaces the IPv6 destination address with the next hop found in the routing header and transmits the message over the OMNI interface identified by the Anycast ULA.

Multiple distinct OMNI links can therefore be used to support fault tolerance, load balancing, reliability, etc. The architectural model is similar to Layer 2 Virtual Local Area Networks (VLANs).

12.2. MN<->AR Traffic Loop Prevention

After an AR has registered an MNP for a MN (see: Section 13), the AR will forward packets destined to an address within the MNP to the MN. The MN will under normal circumstances then forward the packet to the correct destination within its internal networks.

If at some later time the MN loses state (e.g., after a reboot), it may begin returning packets destined to an MNP address to the AR as its default router. The AR therefore must drop any packets originating from the MN and destined to an address within the MN's registered MNP. To do so, the AR institutes the following check:

- o if the IP destination address belongs to a neighbor on the same OMNI interface, and if the link-layer source address is the same as one of the neighbor's link-layer addresses, drop the packet.

13. Router Discovery and Prefix Registration

MNs interface with the MS by sending RS messages with OMNI options under the assumption that one or more AR on the *NET will process the message and respond. The MN then configures default routes for the OMNI interface via the discovered ARs as the next hop. The manner in which the *NET ensures AR coordination is link-specific and outside the scope of this document (however, considerations for *NETs that do not provide ARs that recognize the OMNI option are discussed in Section 18).

For each underlying interface, the MN sends an RS message with an OMNI option to coordinate with MSEs identified by MSID values. Example MSID discovery methods are given in [RFC5214] and include data link login parameters, name service lookups, static configuration, a static "hosts" file, etc. The MN can also send an RS with an MS-Register sub-option that includes the Anycast MSID value '0', i.e., instead of or in addition to any non-zero MSIDs. When the AR receives an RS with a MSID '0', it selects a nearby MSE (which may be itself) and returns an RA with the selected MSID in an MS-Register sub-option. The AR selects only a single wildcard MSE (i.e., even if the RS MS-Register sub-option included multiple '0' MSIDs) while also soliciting the MSEs corresponding to any non-zero MSIDs.

MNs configure OMNI interfaces that observe the properties discussed in the previous section. The OMNI interface and its underlying interfaces are said to be in either the "UP" or "DOWN" state according to administrative actions in conjunction with the interface connectivity status. An OMNI interface transitions to UP or DOWN through administrative action and/or through state transitions of the underlying interfaces. When a first underlying interface transitions to UP, the OMNI interface also transitions to UP. When all underlying interfaces transition to DOWN, the OMNI interface also transitions to DOWN.

When an OMNI interface transitions to UP, the MN sends RS messages to register its MNP and an initial set of underlying interfaces that are also UP. The MN sends additional RS messages to refresh lifetimes and to register/deregister underlying interfaces as they transition to UP or DOWN. The MN sends initial RS messages over an UP underlying interface with its MNP-LLA as the source and with destination set to All-Routers multicast (ff02::2) [RFC4291]. The RS messages include an OMNI option per Section 10 with a Preflen assertion, Interface Attributes appropriate for underlying interfaces, MS-Register/Release sub-options containing MSID values, and with any other necessary OMNI sub-options (e.g., a DUID sub-option as an identity for the MN). The S/T-ifIndex field is set to the index of the underlying interface over which the RS message is sent.

ARs process IPv6 ND messages with OMNI options and act as an MSE themselves and/or as a proxy for other MSEs. ARs receive RS messages and create a neighbor cache entry for the MN, then coordinate with any MSEs named in the Register/Release lists in a manner outside the scope of this document. When an MSE processes the OMNI information, it first validates the prefix registration information then injects/withdraws the MNP in the routing/mapping system and caches/discards the new Preflen, MNP and Interface Attributes. The MSE then informs the AR of registration success/failure, and the AR returns an RA message to the MN with an OMNI option per Section 10.

The AR returns the RA message via the same underlying interface of the MN over which the RS was received, and with destination address set to the MNP-LLA (i.e., unicast), with source address set to its own LLA, and with an OMNI option with S/T-ifIndex set to the value included in the RS. The OMNI option also includes a Preflen confirmation, Interface Attributes, MS-Register/Release and any other necessary OMNI sub-options (e.g., a DUID sub-option as an identity for the AR). The RA also includes any information for the link, including RA Cur Hop Limit, M and O flags, Router Lifetime, Reachable Time and Retrans Timer values, and includes any necessary options such as:

o PIOs with (A; L=0) that include MSPs for the link [RFC8028].

o RIOs [RFC4191] with more-specific routes.

o an MTU option that specifies the maximum acceptable packet size for this underlying interface.

The AR MAY also send periodic and/or event-driven unsolicited RA messages per [RFC4861]. In that case, the S/T-ifIndex field in the OMNI header of the unsolicited RA message identifies the target underlying interface of the destination MN.

The AR can combine the information from multiple MSEs into one or more "aggregate" RAs sent to the MN in order conserve *NET bandwidth. Each aggregate RA includes an OMNI option with MS-Register/Release sub-options with the MSEs represented by the aggregate. If an aggregate is sent, the RA message contents must consistently represent the combined information advertised by all represented MSEs. Note that since the AR uses its own ADM-LLA as the RA source address, the MN determines the addresses of the represented MSEs by examining the MS-Register/Release OMNI sub-options.

When the MN receives the RA message, it creates an OMNI interface neighbor cache entry for each MSID that has confirmed MNP registration via the L2 address of this AR. If the MN connects to multiple *NETs, it records the additional L2 AR addresses in each MSID neighbor cache entry (i.e., as multilink neighbors). The MN then configures a default route via the MSE that returned the RA message, and assigns the Subnet Router Anycast address corresponding to the MNP (e.g., 2001:db8:1:2::) to the OMNI interface. The MN then manages its underlying interfaces according to their states as follows:

- When an underlying interface transitions to UP, the MN sends an RS over the underlying interface with an OMNI option. The OMNI option contains at least one Interface Attribute sub-option with values specific to this underlying interface, and may contain additional Interface Attributes specific to other underlying interfaces. The option also includes any MS-Register/Release sub-options.

- When an underlying interface transitions to DOWN, the MN sends an RS or unsolicited NA message over any UP underlying interface with an OMNI option containing an Interface Attribute sub-option for the DOWN underlying interface with Link set to '0'. The MN sends an RS when an acknowledgement is required, or an unsolicited NA when reliability is not thought to be a concern (e.g., if redundant transmissions are sent on multiple underlying interfaces).

- When the Router Lifetime for a specific AR nears expiration, the MN sends an RS over the underlying interface to receive a fresh RA. If no RA is received, the MN can send RS messages to an alternate MSID in case the current MSID has failed. If no RS messages are received even after trying to contact alternate MSIDs, the MN marks the underlying interface as DOWN.

- When a MN wishes to release from one or more current MSIDs, it sends an RS or unsolicited NA message over any UP underlying interfaces with an OMNI option with a Release MSID. Each MSID then withdraws the MNP from the routing/mapping system and informs the AR that the release was successful.

- When all of a MNs underlying interfaces have transitioned to DOWN (or if the prefix registration lifetime expires), any associated MSEs withdraw the MNP the same as if they had received a message with a release indication.

The MN is responsible for retrying each RS exchange up to MAX_RTR_SOLICITATIONS times separated by RTR_SOLICITATION_INTERVAL seconds until an RA is received. If no RA is received over an UP underlying interface (i.e., even after attempting to contact alternate MSEs), the MN declares this underlying interface as DOWN.

The IPv6 layer sees the OMNI interface as an ordinary IPv6 interface. Therefore, when the IPv6 layer sends an RS message the OMNI interface returns an internally-generated RA message as though the message originated from an IPv6 router. The internally-generated RA message contains configuration information that is consistent with the information received from the RAs generated by the MS. Whether the OMNI interface IPv6 ND messaging process is initiated from the receipt of an RS message from the IPv6 layer is an implementation matter. Some implementations may elect to defer the IPv6 ND messaging process until an RS is received from the IPv6 layer, while others may elect to initiate the process proactively. Still other deployments may elect to administratively disable the ordinary RS/RA messaging used by the IPv6 layer over the OMNI interface, since they are not required to drive the internal RS/RA processing. (Note that this same logic applies to IPv4 implementations that employ ICMP-based Router Discovery per [RFC1256].)

Note: The Router Lifetime value in RA messages indicates the time before which the MN must send another RS message over this underlying interface (e.g., 600 seconds), however that timescale may be significantly longer than the lifetime the MS has committed to retain the prefix registration (e.g., REACHABLETIME seconds). ARs are therefore responsible for keeping MS state alive on a shorter timescale than the MN is required to do on its own behalf.

Note: On multicast-capable underlying interfaces, MNs should send periodic unsolicited multicast NA messages and ARs should send periodic unsolicited multicast RA messages as "beacons" that can be heard by other nodes on the link. If a node fails to receive a beacon after a timeout value specific to the link, it can initiate a unicast exchange to test reachability.

Note: if an AR acting as a proxy forwards a MN's RS message to another node acting as an MSE using UDP/IP encapsulation, it must use a distinct UDP source port number for each MN. This allows the MSE to distinguish different MNs behind the same AR at the link-layer, whereas the link-layer addresses would otherwise be indistinguishable.

Note: when an AR acting as an MSE returns an RA to an INET Client, it includes an OMNI option with an Interface Attributes sub-option with ifIndex set to 0 and with SRT, FMT, LHS and L2ADDR information for its INET interface. This provides the Client with partition prefix context regarding the local OMNI link segment.

13.1. Router Discovery in IP Multihop and IPv4-Only Networks

On some *NETs, a MN may be located multiple IP hops away from the nearest AR. Forwarding through IP multihop *NETs is conducted through the application of a routing protocol (e.g., a Mobile Ad-hoc Network (MANET) routing protocol over omni-directional wireless interfaces, an inter-domain routing protocol in an enterprise network, etc.). These *NETs could be either IPv6-enabled or IPv4-only, while IPv4-only *NETs could be either multicast-capable or unicast-only (note that for IPv4-only *NETs the following procedures apply for both single-hop and multihop cases).

A MN located potentially multiple *NET hops away from the nearest AR prepares an RS message with source address set to either its MNP-LLA or a Temporary LLA, and with destination set to link-scoped All-Routers multicast the same as discussed above. For IPv6-enabled *NETs, the MN then encapsulates the message in an IPv6 header with source address set to the ULA corresponding to the LLA source address and with destination set to either a unicast or anycast ADM-ULA. For IPv4-only *NETs, the MN instead encapsulates the RS message in an IPv4 header with source address set to the node's own IPv4 address and with destination address set to either the unicast IPv4 address of an AR [RFC5214] or an IPv4 anycast address reserved for OMNI. The MN then sends the encapsulated RS message via the *NET interface, where it will be forwarded by zero or more intermediate *NET hops.

When an intermediate *NET hop that participates in the routing protocol receives the encapsulated RS, it forwards the message according to its routing tables (note that an intermediate node could be a fixed infrastructure element or another MN). This process repeats iteratively until the RS message is received by a penultimate *NET hop within single-hop communications range of an AR, which forwards the message to the AR.

When the AR receives the message, it decapsulates the RS and coordinates with the MS the same as for an ordinary link-local RS, since the inner Hop Limit will not have been decremented by the multihop forwarding process. The AR then prepares an RA message with source address set to its own ADM-LLA and destination address set to the LLA of the original MN, then encapsulates the message in an IPv4/IPv6 header with source address set to its own IPv4/ULA address and with destination set to the encapsulation source of the RS.

The AR then forwards the message to an *NET node within communications range, which forwards the message according to its routing tables to an intermediate node. The multihop forwarding process within the *NET continues repetitively until the message is delivered to the original MN, which decapsulates the message and performs autoconfiguration the same as if it had received the RA directly from the AR as an on-link neighbor.

Note: An alternate approach to multihop forwarding via IPv6 encapsulation would be to statelessly translate the IPv6 LLAs into ULAs and forward the messages without encapsulation. This would violate the [RFC4861] requirement that certain IPv6 ND messages must use link-local addresses and must not be accepted if received with Hop Limit less than 255. This document therefore advocates encapsulation since the overhead is nominal considering the infrequent nature and small size of IPv6 ND messages. Future documents may consider encapsulation avoidance through translation while updating [RFC4861].

Note: An alternate approach to multihop forwarding via IPv4 encapsulation would be to employ IPv6/IPv4 protocol translation. However, for IPv6 ND messages the LLAs would be truncated due to translation and the OMNI Router and Prefix Discovery services would not be able to function. The use of IPv4 encapsulation is therefore indicated.

Note: An IPv4 anycast address for OMNI in IPv4 networks could be part of a new IPv4 /24 prefix allocation, but this may be difficult to obtain given IPv4 address exhaustion. An alternative would be to re-purpose the prefix 192.88.99.0 which has been set aside from its former use by [RFC7526].

13.2. MS-Register and MS-Release List Processing

When a MN sends an RS message with an OMNI option via an underlying interface to an AR, the MN must convey its knowledge of its currently-associated MSEs. Initially, the MN will have no associated MSEs and should therefore include an MS-Register sub-option with the single MSID value 0 which requests the AR to select and assign an MSE. The AR will then return an RA message with source address set to the ADM-LLA of the selected MSE.

As the MN activates additional underlying interfaces, it can optionally include an MS-Register sub-option with MSID value 0, or with non-zero MSIDs for MSEs discovered from previous RS/RA exchanges. The MN will thus eventually begin to learn and manage its currently active set of MSEs, and can register with new MSEs or release from former MSEs with each successive RS/RA exchange. As the MN’s MSE constituency grows, it alone is responsible for including or omitting MSIDs in the MS-Register/Release lists it sends in RS messages. The inclusion or omission of MSIDs determines the MN’s interface to the MS and defines the manner in which MSEs will respond. The only limiting factor is that the MN should include no more than MAX_MSID values in each list per each IPv6 ND message, and should avoid duplication of entries in each list unless it wants to increase likelihood of control message delivery.

When an AR receives an RS message sent by a MN with an OMNI option, the option will contain zero or more MS-Register and MS-Release sub-options containing MSIDs. After processing the OMNI option, the AR will have a list of zero or more MS-Register MSIDs and a list of zero or more of MS-Release MSIDs. The AR then processes the lists as follows:

o For each list, retain the first MAX_MSID values in the list and discard any additional MSIDs (i.e., even if there are duplicates within a list).

o Next, for each MSID in the MS-Register list, remove all matching MSIDs from the MS-Release list.

o Next, proceed according to whether the AR's own MSID or the value 0 appears in the MS-Register list as follows:

   * If yes, send an RA message directly back to the MN and send a proxy copy of the RS message to each additional MSID in the MS-Register list with the MS-Register/Release lists omitted. Then, send a uNA message to each MSID in the MS-Release list with the MS-Register/Release lists omitted and with an OMNI header with S/T-ifIndex set to 0.

   * If no, send a proxy copy of the RS message to each additional MSID in the MS-Register list with the MS-Register list omitted. For the first MSID, include the original MS-Release list; for all other MSIDs, omit the MS-Release list.

Each proxy copy of the RS message will include an OMNI option and encapsulation header with the ADM-ULA of the AR as the source and the ADM-ULA of the Register MSE as the destination. When the Register MSE receives the proxy RS message, if the message includes an MS-Release list the MSE sends a uNA message to each additional MSID in the Release list. The Register MSE then sends an RA message back to the (Proxy) AR wrapped in an OMNI encapsulation header with source and destination addresses reversed, and with RA destination set to the MNP-LLA of the MN. When the AR receives this RA message, it sends a proxy copy of the RA to the MN.

Each uNA message (whether send by the first-hop AR or by a Register MSE) will include an OMNI option and an encapsulation header with the ADM-ULA of the Register MSE as the source and the ADM-ULA of the Release ME as the destination. The uNA informs the Release MSE that its previous relationship with the MN has been released and that the source of the uNA message is now registered. The Release MSE must then note that the subject MN of the uNA message is now "departed", and forward any subsequent packets destined to the MN to the Register MSE.

Note that it is not an error for the MS-Register/Release lists to include duplicate entries. If duplicates occur within a list, the AR will generate multiple proxy RS and/or uNA messages - one for each copy of the duplicate entries.

13.3. DHCPv6-based Prefix Registration

When a MN is not pre-provisioned with an MNP-LLA (or, when multiple MNPs are needed), it will require the AR to select MNPs on its behalf and set up the correct routing state within the MS. The DHCPv6 service [RFC8415] supports this requirement.

When an MN needs to have the AR select MNPs, it sends an RS message with a Temporary LLA as the source and with DHCPv6 Message sub-option containing a Client Identifier, one or more IA_PD options and a Rapid Commit option. The MN also sets the 'msg-type' field to "Solicit", and includes a 3-octet 'transaction-id'.

When the AR receives the RS message, it extracts the DHCPv6 message from the OMNI option. The AR then acts as a "Proxy DHCPv6 Client" in a message exchange with the locally-resident DHCPv6 server, which delegates MNPs and returns a DHCPv6 Reply message with PD parameters. (If the AR wishes to defer creation of MN state until the DHCPv6 Reply is received, it can instead act as a Lightweight DHCPv6 Relay Agent per [RFC6221] by encapsulating the DHCPv6 message in a Relay-forward/reply exchange with Relay Message and Interface ID options.)

When the AR receives the DHCPv6 Reply, it adds routes to the routing system and creates MNP-LLAs based on the delegated MNPs. The AR then sends an RA back to the MN with the DHCPv6 Reply message included in an OMNI DHCPv6 message sub-option. If the RS message source address was a Temporary address, the AR includes one of the (newly-created) MNP-LLAs as the RA destination address. The MN then creates a default route, assigns Subnet Router Anycast addresses and uses the RA destination address as its primary MNP-LLA. The MN will then use this primary MNP-LLA as the source address of any IPv6 ND messages it sends as long as it retains ownership of the MNP.

Note: The single-octet OMNI sub-option length field restricts the DHCPv6 Message sub-option to a maximum of 255 octets for both the RS and RA messages. This provides sufficient room for the DHCPv6 message header, a Client/Server Identifier option, a Rapid Commit option, at least 3 Identity Association for Prefix Delegation (IA_PD) options and any other supporting DHCPv6 options. A MN requiring more DHCPv6-based configuration information than this can either perform multiple independent RS/RA exchanges (with each exchange providing a subset of the total configuration information) or simply perform an actual DHCPv6 message exchange in addition to any RS/RA exchanges.

Note: After a MN performs a DHCPv6-based prefix registration exchange with a first AR, it would need to repeat the exchange with each additional MSE it registers with. In that case, the MN supplies the MNP delegations received from the first AR in the IA_PD fields of a DHCPv6 message when it engages the additional MSEs.

14. Secure Redirection

If the *NET link model is multiple access, the AR is responsible for assuring that address duplication cannot corrupt the neighbor caches of other nodes on the link. When the MN sends an RS message on a multiple access *NET link, the AR verifies that the MN is authorized to use the address and returns an RA with a non-zero Router Lifetime only if the MN is authorized.

After verifying MN authorization and returning an RA, the AR MAY return IPv6 ND Redirect messages to direct MNs located on the same *NET link to exchange packets directly without transiting the AR. In that case, the MNs can exchange packets according to their unicast L2 addresses discovered from the Redirect message instead of using the dogleg path through the AR. In some *NET links, however, such direct communications may be undesirable and continued use of the dogleg path through the AR may provide better performance. In that case, the AR can refrain from sending Redirects, and/or MNs can ignore them.

15. AR and MSE Resilience

*NETs SHOULD deploy ARs in Virtual Router Redundancy Protocol (VRRP) [RFC5798] configurations so that service continuity is maintained even if one or more ARs fail. Using VRRP, the MN is unaware which of the (redundant) ARs is currently providing service, and any service discontinuity will be limited to the failover time supported by VRRP. Widely deployed public domain implementations of VRRP are available.

MSEs SHOULD use high availability clustering services so that multiple redundant systems can provide coordinated response to failures. As with VRRP, widely deployed public domain implementations of high availability clustering services are available. Note that special-purpose and expensive dedicated hardware is not necessary, and public domain implementations can be used even between lightweight virtual machines in cloud deployments.

16. Detecting and Responding to MSE Failures

In environments where fast recovery from MSE failure is required, ARs SHOULD use proactive Neighbor Unreachability Detection (NUD) in a manner that parallels Bidirectional Forwarding Detection (BFD)

[RFC5880] to track MSE reachability. ARs can then quickly detect and react to failures so that cached information is re-established through alternate paths. Proactive NUD control messaging is carried only over well-connected ground domain networks (i.e., and not low-end *NET links such as aeronautical radios) and can therefore be tuned for rapid response.

ARs perform proactive NUD for MSEs for which there are currently active MNs on the *NET. If an MSE fails, ARs can quickly inform MNs of the outage by sending multicast RA messages on the *NET interface. The AR sends RA messages to MNs via the *NET interface with an OMNI option with a Release ID for the failed MSE, and with destination address set to All-Nodes multicast (ff02::1) [RFC4291].

The AR SHOULD send MAX_FINAL_RTR_ADVERTISEMENTS RA messages separated by small delays [RFC4861]. Any MNs on the *NET interface that have been using the (now defunct) MSE will receive the RA messages and associate with a new MSE.

17. Transition Considerations

When a MN connects to an *NET link for the first time, it sends an RS message with an OMNI option. If the first hop AR recognizes the option, it returns an RA with its ADM-LLA as the source, the MNP-LLA as the destination and with an OMNI option included. The MN then engages the AR according to the OMNI link model specified above. If the first hop AR is a legacy IPv6 router, however, it instead returns an RA message with no OMNI option and with a non-OMNI unicast source LLA as specified in [RFC4861]. In that case, the MN engages the *NET according to the legacy IPv6 link model and without the OMNI extensions specified in this document.

If the *NET link model is multiple access, there must be assurance that address duplication cannot corrupt the neighbor caches of other nodes on the link. When the MN sends an RS message on a multiple access *NET link with an LLA source address and an OMNI option, ARs that recognize the option ensure that the MN is authorized to use the address and return an RA with a non-zero Router Lifetime only if the MN is authorized. ARs that do not recognize the option instead return an RA that makes no statement about the MN's authorization to use the source address. In that case, the MN should perform Duplicate Address Detection to ensure that it does not interfere with other nodes on the link.

An alternative approach for multiple access *NET links to ensure isolation for MN / AR communications is through L2 address mappings as discussed in Appendix C. This arrangement imparts a (virtual) point-to-point link model over the (physical) multiple access link.

18. OMNI Interfaces on the Open Internet

OMNI interfaces configured over IPv6-enabled underlying interfaces on the open Internet without an OMNI-aware first-hop AR receive RA messages that do not include an OMNI option, while OMNI interfaces configured over IPv4-only underlying interfaces do not receive any (IPv6) RA messages at all. OMNI interfaces that receive RA messages without an OMNI option configure addresses, on-link prefixes, etc. on the underlying interface that received the RA according to standard IPv6 ND and address resolution conventions [RFC4861] [RFC4862]. OMNI interfaces configured over IPv4-only underlying interfaces configure IPv4 address information on the underlying interfaces using mechanisms such as DHCPv4 [RFC2131].

OMNI interfaces configured over underlying interfaces that connect to the open Internet can apply security services such as VPNs to connect to an MSE, or can establish a direct link to an MSE through some other means (see Section 4). In environments where an explicit VPN or direct link may be impractical, OMNI interfaces can instead use UDP/IP encapsulation and HMAC-based message authentication per [RFC6081][RFC4380].

After establishing a VPN or preparing for UDP/IP encapsulation, OMNI interfaces send control plane messages to interface with the MS, including RS/RA messages used according to Section 13 and Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages used for address resolution / route optimization (see: [I-D.templin-intarea-6706bis]). The control plane messages must be authenticated while data plane messages are delivered the same as for ordinary best-effort Internet traffic with basic source address-based data origin verification. Data plane communications via OMNI interfaces that connect over the open Internet without an explicit VPN should therefore employ transport- or higher-layer security to ensure integrity and/or confidentiality.

OMNI interfaces in the open Internet are often located behind NATs. The OMNI interface accommodates NAT traversal using UDP/IP encapsulation and the mechanisms discussed in [RFC6081][RFC4380][I-D.templin-intarea-6706bis].

19. Time-Varying MNPs

In some use cases, it is desirable, beneficial and efficient for the MN to receive a constant MNP that travels with the MN wherever it moves. For example, this would allow air traffic controllers to easily track aircraft, etc. In other cases, however (e.g., intelligent transportation systems), the MN may be willing to sacrifice a modicum of efficiency in order to have time-varying MNPs that can be changed every so often to defeat adversarial tracking.

The prefix delegation services discussed in Section 13.3 allows OMNI MNs that desire time-varying MNPs to obtain short-lived prefixes to use a Temporary LLA as the source address of an RS message with an OMNI option with DHCPv6 Option sub-options. The MN would then be obligated to renumber its internal networks whenever its MNP (and therefore also its OMNI address) changes. This should not present a challenge for MNs with automated network renumbering services, however presents limits for the durations of ongoing sessions that would prefer to use a constant address.

20. IANA Considerations

The IANA is instructed to allocate an official Type number TBD from the registry "IPv6 Neighbor Discovery Option Formats" for the OMNI option. Implementations set Type to 253 as an interim value [RFC4727].

The IANA is instructed to assign a new Code value "1" in the "ICMPv6 Code Fields: Type 2 - Packet Too Big" registry. The registry should read as follows:

Code	Name	Reference
0	Diagnostic Packet Too Big	[RFC4443]
1	Advisory Packet Too Big	[RFCXXXX]

Figure 17: ICMPv6 Code Fields: Type 2 - Packet Too Big Values

The IANA is instructed to allocate one Ethernet unicast address TBD2 (suggest 00-00-5E-00-52-14 [RFC5214]) in the registry "IANA Ethernet Address Block - Unicast Use".

The OMNI option also defines an 8-bit Sub-Type field, for which IANA is instructed to create and maintain a new registry entitled "OMNI option Sub-Type values". Initial values for the OMNI option Sub-Type values registry are given below; future assignments are to be made through Expert Review [RFC8126].

Value	Sub-Type name	Reference
0	Pad1	[RFCXXXX]
1	PadN	[RFCXXXX]
2	Interface Attributes	[RFCXXXX]
3	Traffic Selector	[RFCXXXX]
4	MS-Register	[RFCXXXX]
5	MS-Release	[RFCXXXX]
6	Network Access Identifier	[RFCXXXX]
7	Geo Coordinates	[RFCXXXX]
8	DHCP Unique Identifier (DUID)	[RFCXXXX]
9	DHCPv6 Message	[RFCXXXX]
10-252	Unassigned	
253-254	Experimental	[RFCXXXX]
255	Reserved	[RFCXXXX]

Figure 18: OMNI Option Sub-Type Values

21. Security Considerations

Security considerations for IPv4 [RFC0791], IPv6 [RFC8200] and IPv6 Neighbor Discovery [RFC4861] apply. OMNI interface IPv6 ND messages SHOULD include Nonce and Timestamp options [RFC3971] when transaction confirmation and/or time synchronization is needed.

OMNI interfaces configured over secured ANET interfaces inherit the physical and/or link-layer security properties (i.e., "protected spectrum") of the connected ANETs. OMNI interfaces configured over open INET interfaces can use symmetric securing services such as VPNs or can by some other means establish a direct link. When a VPN or direct link may be impractical, however, an asymmetric security service such as the authentication option specified in [RFC4380] or other protocol control message security mechanisms may be necessary. While the OMNI link protects control plane messaging, applications must still employ end-to-end transport- or higher-layer security services to protect the data plane.

The Mobility Service MUST provide strong network layer security for control plane messages and forwarding path integrity for data plane messages. In one example, the AERO service [I-D.templin-intarea-6706bis] constructs a spanning tree between mobility service elements and secures the links in the spanning tree with network layer security mechanisms such as IPsec [RFC4301] or Wireguard. Control plane messages are then constrained to travel only over the secured spanning tree paths and are therefore protected from attack or eavesdropping. Since data plane messages can travel over route optimized paths that do not strictly follow the spanning tree, however, end-to-end transport- or higher-layer security services are still required.

Security considerations for specific access network interface types are covered under the corresponding IP-over-(foo) specification (e.g., [RFC2464], [RFC2492], etc.).

Security considerations for IPv6 fragmentation and reassembly are discussed in Section 5.1.

22. Implementation Status

Draft -29 is implemented in the recently tagged AERO/OMNI 3.0.0 internal release, and Draft -30 is now tagged as the AERO/OMNI 3.0.1. Newer specification versions will be tagged in upcoming releases. First public release expected before the end of 2020.

23. Acknowledgements

The first version of this document was prepared per the consensus decision at the 7th Conference of the International Civil Aviation Organization (ICAO) Working Group-I Mobility Subgroup on March 22, 2019. Consensus to take the document forward to the IETF was reached at the 9th Conference of the Mobility Subgroup on November 22, 2019. Attendees and contributors included: Guray Acar, Danny Bharj, Francois D'Humieres, Pavel Drasil, Nikos Fistas, Giovanni Garofolo, Bernhard Haindl, Vaughn Maiolla, Tom McParland, Victor Moreno, Madhu Niraula, Brent Phillips, Liviu Popescu, Jacky Pouzet, Aloke Roy, Greg Saccone, Robert Segers, Michal Skorepa, Michel Solery, Stephane Tamalet, Fred Templin, Jean-Marc Vacher, Bela Varkonyi, Tony Whyman, Fryderyk Wrobel and Dongsong Zeng.

The following individuals are acknowledged for their useful comments: Michael Matyas, Madhu Niraula, Greg Saccone, Stephane Tamalet, Eric Vyncke. Pavel Drasil, Zdenek Jaron and Michal Skorepa are recognized for their many helpful ideas and suggestions. Madhuri Madhava Badgandi, Katherine Tran, and Vijayasarathy Rajagopalan are acknowledged for their hard work on the implementation and insights that led to improvements to the spec.

Discussions on the IETF 6man and atn mailing lists during the fall of 2020 suggested additional points to consider. The authors gratefully acknowledge the list members who contributed valuable insights through those discussions. Eric Vyncke and Erik Kline were the intarea ADs, while Bob Hinden and Ole Troan were the 6man WG chairs at the time the document was developed; they are all gratefully acknowledged for their many helpful insights.

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

24. References

24.1. Normative References

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

[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>.

[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, <https://www.rfc-editor.org/info/rfc2474>.

[RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <https://www.rfc-editor.org/info/rfc3971>.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <https://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, <https://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, <https://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", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, DOI 10.17487/RFC4727, November 2006, <https://www.rfc-editor.org/info/rfc4727>.

[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, <https://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, <https://www.rfc-editor.org/info/rfc4862>.

[RFC6088] Tsirtsis, G., Giarreta, G., Soliman, H., and N. Montavont, "Traffic Selectors for Flow Bindings", RFC 6088, DOI 10.17487/RFC6088, January 2011, <https://www.rfc-editor.org/info/rfc6088>.

[RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <https://www.rfc-editor.org/info/rfc8028>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

[RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <https://www.rfc-editor.org/info/rfc8415>.

24.2. Informative References

[ATN] Maiolla, V., "The OMNI Interface - An IPv6 Air/Ground Interface for Civil Aviation, IETF Liaison Statement #1676, https://datatracker.ietf.org/liaison/1676/", March 2020.

[ATN-IPS] WG-I, ICAO., "ICAO Document 9896 (Manual on the Aeronautical Telecommunication Network (ATN) using Internet Protocol Suite (IPS) Standards and Protocol), Draft Edition 3 (work-in-progress)", December 2020.

[CRC] Jain, R., "Error Characteristics of Fiber Distributed Data Interface (FDDI), IEEE Transactions on Communications", August 1990.

[I-D.ietf-6man-rfc4941bis]
Gont, F., Krishnan, S., Narten, T., and R. Draves, "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6", draft-ietf-6man-rfc4941bis-12 (work in progress), November 2020.

[I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-10 (work in progress), September 2019.

[I-D.ietf-ipwave-vehicular-networking]
Jeong, J., "IPv6 Wireless Access in Vehicular Environments (IPWAVE): Problem Statement and Use Cases", draft-ietf-ipwave-vehicular-networking-19 (work in progress), July 2020.

[I-D.templin-6man-dhcpv6-ndopt]
Templin, F., "A Unified Stateful/Stateless Configuration Service for IPv6", draft-templin-6man-dhcpv6-ndopt-10 (work in progress), June 2020.

[I-D.templin-6man-lla-type]
Templin, F., "The IPv6 Link-Local Address Type Field", draft-templin-6man-lla-type-02 (work in progress), November 2020.

[I-D.templin-intarea-6706bis]
Templin, F., "Asymmetric Extended Route Optimization (AERO)", draft-templin-intarea-6706bis-86 (work in progress), December 2020.

[IPV4-GUA]
Postel, J., "IPv4 Address Space Registry, https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml", December 2020.

[IPV6-GUA]
Postel, J., "IPv6 Global Unicast Address Assignments, https://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml", December 2020.

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

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

[RFC1256] Deering, S., Ed., "ICMP Router Discovery Messages", RFC 1256, DOI 10.17487/RFC1256, September 1991, <https://www.rfc-editor.org/info/rfc1256>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <https://www.rfc-editor.org/info/rfc2131>.

[RFC2225] Laubach, M. and J. Halpern, "Classical IP and ARP over ATM", RFC 2225, DOI 10.17487/RFC2225, April 1998, <https://www.rfc-editor.org/info/rfc2225>.

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

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

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

[RFC2529] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529, DOI 10.17487/RFC2529, March 1999, <https://www.rfc-editor.org/info/rfc2529>.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <https://www.rfc-editor.org/info/rfc2863>.

[RFC3330] IANA, "Special-Use IPv4 Addresses", RFC 3330, DOI 10.17487/RFC3330, September 2002, <https://www.rfc-editor.org/info/rfc3330>.

[RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, DOI 10.17487/RFC3692, January 2004, <https://www.rfc-editor.org/info/rfc3692>.

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

[RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <https://www.rfc-editor.org/info/rfc3819>.

[RFC3879] Huitema, C. and B. Carpenter, "Deprecating Site Local Addresses", RFC 3879, DOI 10.17487/RFC3879, September 2004, <https://www.rfc-editor.org/info/rfc3879>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

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

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

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <https://www.rfc-editor.org/info/rfc4389>.

[RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, DOI 10.17487/RFC4429, April 2006, <https://www.rfc-editor.org/info/rfc4429>.

[RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <https://www.rfc-editor.org/info/rfc4541>.

[RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, DOI 10.17487/RFC4605, August 2006, <https://www.rfc-editor.org/info/rfc4605>.

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

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <https://www.rfc-editor.org/info/rfc4963>.

[RFC5175] Haberman, B., Ed. and R. Hinden, "IPv6 Router Advertisement Flags Option", RFC 5175, DOI 10.17487/RFC5175, March 2008, <https://www.rfc-editor.org/info/rfc5175>.

[RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, DOI 10.17487/RFC5213, August 2008, <https://www.rfc-editor.org/info/rfc5213>.

[RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, DOI 10.17487/RFC5214, March 2008, <https://www.rfc-editor.org/info/rfc5214>.

[RFC5558] Templin, F., Ed., "Virtual Enterprise Traversal (VET)", RFC 5558, DOI 10.17487/RFC5558, February 2010, <https://www.rfc-editor.org/info/rfc5558>.

[RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <https://www.rfc-editor.org/info/rfc5798>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC6081] Thaler, D., "Teredo Extensions", RFC 6081, DOI 10.17487/RFC6081, January 2011, <https://www.rfc-editor.org/info/rfc6081>.

[RFC6221] Miles, D., Ed., Ooghe, S., Dec, W., Krishnan, S., and A. Kavanagh, "Lightweight DHCPv6 Relay Agent", RFC 6221, DOI 10.17487/RFC6221, May 2011, <https://www.rfc-editor.org/info/rfc6221>.

[RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, DOI 10.17487/RFC6355, August 2011, <https://www.rfc-editor.org/info/rfc6355>.

[RFC6543] Gundavelli, S., "Reserved IPv6 Interface Identifier for Proxy Mobile IPv6", RFC 6543, DOI 10.17487/RFC6543, May 2012, <https://www.rfc-editor.org/info/rfc6543>.

[RFC7084] Singh, H., Beebee, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, DOI 10.17487/RFC7084, November 2013, <https://www.rfc-editor.org/info/rfc7084>.

[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, <https://www.rfc-editor.org/info/rfc7421>.

[RFC7526] Troan, O. and B. Carpenter, Ed., "Deprecating the Anycast Prefix for 6to4 Relay Routers", BCP 196, RFC 7526, DOI 10.17487/RFC7526, May 2015, <https://www.rfc-editor.org/info/rfc7526>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <https://www.rfc-editor.org/info/rfc7542>.

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

[RFC7847] Melia, T., Ed. and S. Gundavelli, Ed., "Logical-Interface Support for IP Hosts with Multi-Access Support", RFC 7847, DOI 10.17487/RFC7847, May 2016, <https://www.rfc-editor.org/info/rfc7847>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

[RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <https://www.rfc-editor.org/info/rfc8900>.

Appendix A. Interface Attribute Preferences Bitmap Encoding

Adaptation of the OMNI option Interface Attributes Preferences Bitmap encoding to specific Internetworks such as the Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS) may include link selection preferences based on other traffic classifiers (e.g., transport port numbers, etc.) in addition to the existing DSCP-based preferences. Nodes on specific Internetworks maintain a map of traffic classifiers to additional P[*] preference fields beyond the first 64. For example, TCP port 22 maps to P[67], TCP port 443 maps to P[70], UDP port 8060 maps to P[76], etc.

Implementations use Simplex or Indexed encoding formats for P[*] encoding in order to encode a given set of traffic classifiers in the most efficient way. Some use cases may be more efficiently coded using Simplex form, while others may be more efficient using Indexed. Once a format is selected for preparation of a single Interface Attribute the same format must be used for the entire Interface Attribute sub-option. Different sub-options may use different formats.

The following figures show coding examples for various Simplex and Indexed formats:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Type=2  |  Sub-length=N |    ifIndex    |     ifType    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Provider ID  | Link  |R| API | Bitmap(0)=0xff|P00|P01|P02|P03|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P04|P05|P06|P07|P08|P09|P10|P11|P12|P13|P14|P15|P16|P17|P18|P19|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P20|P21|P22|P23|P24|P25|P26|P27|P28|P29|P30|P31| Bitmap(1)=0xff|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P32|P33|P34|P35|P36|P37|P38|P39|P40|P41|P42|P43|P44|P45|P46|P47|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P48|P49|P50|P51|P52|P53|P54|P55|P56|P57|P58|P59|P60|P61|P62|P63|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Bitmap(2)=0xff|P64|P65|P67|P68| ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 19: Example 1: Dense Simplex Encoding

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Type=2  |  Sub-length=N |    ifIndex    |     ifType    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Provider ID  | Link  |R| API | Bitmap(0)=0x00| Bitmap(1)=0x0f|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P48|P49|P50|P51|P52|P53|P54|P55|P56|P57|P58|P59|P60|P61|P62|P63|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Bitmap(2)=0x00| Bitmap(3)=0x00| Bitmap(4)=0x00| Bitmap(5)=0x00|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Bitmap(6)=0xf0|192|193|194|195|196|197|198|199|200|201|202|203|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|204|205|206|207| Bitmap(7)=0x00| Bitmap(8)=0x0f|272|273|274|275|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|276|277|278|279|280|281|282|283|284|285|286|287| Bitmap(9)=0x00|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Bitmap(10)=0x00| ...
+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 20: Example 2: Sparse Simplex Encoding

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Type=2  | Sub-length=N  |    ifIndex    |     ifType    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Provider ID  | Link  |R| API |  Index = 0x00 | Bitmap = 0x80 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P00|P01|P02|P03|  Index = 0x01 | Bitmap = 0x01 |P60|P61|P62|P63|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Index = 0x10 | Bitmap = 0x80 |512|513|514|515|  Index = 0x18 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Bitmap = 0x01 |796|797|798|799| ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 21: Example 3: Indexed Encoding

Appendix B. VDL Mode 2 Considerations

ICAO Doc 9776 is the "Technical Manual for VHF Data Link Mode 2" (VDLM2) that specifies an essential radio frequency data link service for aircraft and ground stations in worldwide civil aviation air traffic management. The VDLM2 link type is "multicast capable" [RFC4861], but with considerable differences from common multicast links such as Ethernet and IEEE 802.11.

First, the VDLM2 link data rate is only 31.5Kbps - multiple orders of magnitude less than most modern wireless networking gear. Second, due to the low available link bandwidth only VDLM2 ground stations (i.e., and not aircraft) are permitted to send broadcasts, and even so only as compact layer 2 "beacons". Third, aircraft employ the services of ground stations by performing unicast RS/RA exchanges upon receipt of beacons instead of listening for multicast RA messages and/or sending multicast RS messages.

This beacon-oriented unicast RS/RA approach is necessary to conserve the already-scarce available link bandwidth. Moreover, since the numbers of beaconing ground stations operating within a given spatial range must be kept as sparse as possible, it would not be feasible to have different classes of ground stations within the same region observing different protocols. It is therefore highly desirable that all ground stations observe a common language of RS/RA as specified in this document.

Note that links of this nature may benefit from compression techniques that reduce the bandwidth necessary for conveying the same amount of data. The IETF lpwan working group is considering possible alternatives: [https://datatracker.ietf.org/wg/lpwan/documents].

Appendix C. MN / AR Isolation Through L2 Address Mapping

Per [RFC4861], IPv6 ND messages may be sent to either a multicast or unicast link-scoped IPv6 destination address. However, IPv6 ND messaging should be coordinated between the MN and AR only without invoking other nodes on the *NET. This implies that MN / AR control messaging should be isolated and not overheard by other nodes on the link.

To support MN / AR isolation on some *NET links, ARs can maintain an OMNI-specific unicast L2 address ("MSADDR"). For Ethernet-compatible *NETs, this specification reserves one Ethernet unicast address TBD2 (see: Section 20). For non-Ethernet statically-addressed *NETs, MSADDR is reserved per the assigned numbers authority for the *NET addressing space. For still other *NETs, MSADDR may be dynamically discovered through other means, e.g., L2 beacons.

MNs map the L3 addresses of all IPv6 ND messages they send (i.e., both multicast and unicast) to MSADDR instead of to an ordinary unicast or multicast L2 address. In this way, all of the MN's IPv6 ND messages will be received by ARs that are configured to accept packets destined to MSADDR. Note that multiple ARs on the link could be configured to accept packets destined to MSADDR, e.g., as a basis for supporting redundancy.

Therefore, ARs must accept and process packets destined to MSADDR, while all other devices must not process packets destined to MSADDR. This model has well-established operational experience in Proxy Mobile IPv6 (PMIP) [RFC5213][RFC6543].

Appendix D. Change Log

<< RFC Editor - remove prior to publication >>

Differences from draft-templin-6man-omni-interface-35 to draft-templin-6man-omni-interface-36:

- o Major clarifications on aspects such as "hard/soft" PTB error messages

- o Made generic so that either IP protocol version (IPv4 or IPv6) can be used in the data plane.

Differences from draft-templin-6man-omni-interface-31 to draft-templin-6man-omni-interface-32:

- o MTU

o Support for multi-hop ANETS such as ISATAP.

Differences from draft-templin-6man-omni-interface-29 to draft-templin-6man-omni-interface-30:

o Moved link-layer addressing information into the OMNI option on a per-ifIndex basis

o Renamed "ifIndex-tuple" to "Interface Attributes"

Differences from draft-templin-6man-omni-interface-27 to draft-templin-6man-omni-interface-28:

o Updates based on implementation experience.

Differences from draft-templin-6man-omni-interface-25 to draft-templin-6man-omni-interface-26:

o Further clarification on "aggregate" RA messages.

o Expanded Security Considerations to discuss expectations for security in the Mobility Service.

Differences from draft-templin-6man-omni-interface-20 to draft-templin-6man-omni-interface-21:

o Safety-Based Multilink (SBM) and Performance-Based Multilink (PBM).

Differences from draft-templin-6man-omni-interface-18 to draft-templin-6man-omni-interface-19:

o SEND/CGA.

Differences from draft-templin-6man-omni-interface-17 to draft-templin-6man-omni-interface-18:

o Teredo

Differences from draft-templin-6man-omni-interface-14 to draft-templin-6man-omni-interface-15:

o Prefix length discussions removed.

Differences from draft-templin-6man-omni-interface-12 to draft-templin-6man-omni-interface-13:

o Teredo

Differences from draft-templin-6man-omni-interface-11 to draft-templin-6man-omni-interface-12:

- Major simplifications and clarifications on MTU and fragmentation.

- Document now updates RFC4443 and RFC8201.

Differences from draft-templin-6man-omni-interface-10 to draft-templin-6man-omni-interface-11:

- Removed /64 assumption, resulting in new OMNI address format.

Differences from draft-templin-6man-omni-interface-07 to draft-templin-6man-omni-interface-08:

- OMNI MNs in the open Internet

Differences from draft-templin-6man-omni-interface-06 to draft-templin-6man-omni-interface-07:

- Brought back L2 MSADDR mapping text for MN / AR isolation based on L2 addressing.

- Expanded "Transition Considerations".

Differences from draft-templin-6man-omni-interface-05 to draft-templin-6man-omni-interface-06:

- Brought back OMNI option "R" flag, and discussed its use.

Differences from draft-templin-6man-omni-interface-04 to draft-templin-6man-omni-interface-05:

- Transition considerations, and overhaul of RS/RA addressing with the inclusion of MSE addresses within the OMNI option instead of as RS/RA addresses (developed under FAA SE2025 contract number DTFAWA-15-D-00030).

Differences from draft-templin-6man-omni-interface-02 to draft-templin-6man-omni-interface-03:

- Added "advisory PTB messages" under FAA SE2025 contract number DTFAWA-15-D-00030.

Differences from draft-templin-6man-omni-interface-01 to draft-templin-6man-omni-interface-02:

- Removed "Primary" flag and supporting text.

- Clarified that "Router Lifetime" applies to each ANET interface independently, and that the union of all ANET interface Router Lifetimes determines MSE lifetime.

Differences from draft-templin-6man-omni-interface-00 to draft-templin-6man-omni-interface-01:

- "All-MSEs" OMNI LLA defined. Also reserved fe80::ff00:0000/104 for future use (most likely as "pseudo-multicast").

- Non-normative discussion of alternate OMNI LLA construction form made possible if the 64-bit assumption were relaxed.

First draft version (draft-templin-atn-aero-interface-00):

- Draft based on consensus decision of ICAO Working Group I Mobility Subgroup March 22, 2019.

Authors' Addresses

Fred L. Templin (editor)
The Boeing Company
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

Tony Whyman
MWA Ltd c/o Inmarsat Global Ltd
99 City Road
London EC1Y 1AX
England

Email: tony.whyman@mccallumwhyman.com

BIER
Internet-Draft
Intended status: Standards Track
Expires: June 13, 2021

Z. Zhang
ZTE Corporation
Z. Zhang, Ed.
Juniper Networks
I. Wijnands
Individual
M. Mishra
Cisco Systems
H. Bidgoli
Nokia
G. Mishra, Ed.
Verizon
December 10, 2020

Supporting BIER in IPv6 Networks (BIERin6)
draft-zhang-bier-bierin6-08

Abstract

BIER is a new architecture for the forwarding of multicast data packets without requiring per-flow state inside the network. This document describes how the existing BIER encapsulation specified in RFC 8296 works in an IPv6 non-MPLS network, referred to as BIERin6. Specifically, like in an IPv4 network, BIER can work over L2 links directly or over tunnels. In case of IPv6 tunneling, a new IP "Next Header" type is to be assigned for BIER.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Copyright Notice

Table of Contents

1. Introduction

BIER [RFC8279] is a new architecture for the forwarding of multicast data packets. It provides optimal forwarding through a "multicast domain" and it does not precondition construction of a multicast distribution tree, nor does it require intermediate nodes to maintain any per-flow state.

This document specifies non-MPLS BIER forwarding in an IPv6 [RFC8200] environment, referred to as BIERin6, using non-MPLS BIER encapsulation specified in [RFC8296].

MPLS BIER forwarding in IPv6 is outside the scope of this document.

This document uses terminology defined in [RFC8279] and [RFC8296].

1.1. BIER over L2/Tunnels

[RFC8296] defines the BIER encapsulation format in MPLS and non-MPLS environment. In case of non-MPLS environment, a BIER packet is the payload of an "outer" encapsulation, which has a "next header" codepoint that is set to a value that means "non-MPLS BIER". This "BIER over L2/Tunnel" model can be used as is in an IPv6 non-mpls environment, and is referred to as BIERin6.

If a BFR needs to tunnel BIER packets to another BFR, e.g. per [RFC8279] Section 6.9, while any type of tunnel will work, for best efficiency native IPv6 encapsulation can be used with the destination address being the downstream BFR and the Next Header field set to a to-be-assigned value for "non-MPLS BIER".

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

Between two directly connected BFRs, a BIER header can directly follow link layer header, e.g., an Ethernet header (with the Ethertype set to 0xAB37). Optionally, IPv6 encapsulation can be used even between directly connected BFRs (i.e. one-hop IPv6 tunneling) in the following two cases:

- o An operator mandates all traffic to be carried in IPv6.
- o A BFR does not have BIER support in its "fast forwarding path" and relies on "slow/software forwarding path", e.g. in environments like [RFC7368] where high throughput multicast forwarding performance is not critical.

1.2. Considerations of Requirements for BIER in IPv6 Networks

[draft-ietf-bier-ipv6-requirements] lists mandatory and optional requirements for BIER in IPv6 Networks. As a solution based on the

BIER over L2/tunnel model [RFC8296], BIERin6 satisfies all the mandatory requirements.

For the two optional requirements for fragmentation and Encapsulating Security Payload (ESP), they can be satisfied by one of two ways:

- o IPv6 based fragmentation/ESP: a BFIR encapsulates the payload in IPv6 with fragmentation and/or ESP header, and then the IPv6 packets are treated as BIER payload.

- o Generic Fragmentation/ESP [I-D.zzhang-tsvwg-generic-transport-functions]: a BFIR does generic fragmentation and/or ESP (without using IPv6 encapsulation) and the resulting packets are treated as BIER payload.

Either way, the fragmentation/ESP is handled by a layer outside of BIER and then the resulting packets are treated as BIER payload.

BIERin6 does support SRv6 BGP based overlay services (e.g. MVPN/ EVPN) as following. An ingress PE (which is a BFIR) encapsulates customer packets with an IPv6 header (with optional fragmentation and ESP extension headers). The destination address is a multicast locator plus the Fucn/Arg portion that identifies the VPN. That IPv6 packet is then treated as BIER payload. An egress PE (which is a BFER) uses the standard SRv6 procedures to forward the IPv6 packet that is exposed after the BIER header is decapsulated. Relevant overlay signaling will be specified separately.

BIERin6 being a solution based on [RFC8279] [RFC8296], ECMP is inherently supported by BFRs using the the 20-bit entropy field in the BIER header for the load balancing hash. When a BIER packet is transported over an IPv6 tunnel, the entropy value is copied into the 20-bit IPv6 Flow Label (instead of using local 5-tuple input key to a hash function to locally generate the stateless 20-bit flow label) so that routers along the tunnel can do ECMP based on Flow Labels. For a router along the tunnel doing deep packet inspection for ECMP purpose, if it understands BIER header it can go past the BIER header to look for the 5-tuple input key to a hash function, otherwise it stops at the BIER header. In either case the router will not mistake the BIER header as an IP header so no misordering should happen.

BIER has its own OAM functions independent of those related to the underlying links or tunnels. With BIERin6 following the "BIER over L2/tunnel" model, IPv6 OAM function and BIER OAM functions are used independently for their own purposes.

Specifically, BIERin6 works with all of the following OAM methods, or any future methods that are based on the "BIER over L2/tunnel" model:

- o BIER OAM specified in [I-D.ietf-bier-ping]
- o BIER BFD specified in [I-D.ietf-bier-bfd]
- o BIER Performance Measurement specified in [I-D.ietf-bier-pmmm-oam]
- o BIER Path Maximum Transmission Unit Discovery specified in [I-D.ietf-bier-path-mtu-discovery]
- o BIER IOAM specified in [I-D.xzlnp-bier-ioam]

2. IPv6 Header

Whenever IPv6 encapsulation is used for BIER forwarding, The Next Header field in the IPv6 Header (if there are no extension headers), or the Next Header field in the last extension header is set to TBD, indicating that the payload is a BIER packet.

If the neighbor is directly connected, The destination address in IPv6 header SHOULD be the neighbor's link-local address on this router's outgoing interface, the source destination address SHOULD be this router's link-local address on the outgoing interface, and the IPv6 TTL MUST be set to 1. Otherwise, the destination address SHOULD be the BIER prefix of the BFR neighbor, the source address SHOULD be this router's BIER prefix, and the TTL MUST be large enough to get the packet to the BFR neighbor.

The "Flow label" field in the IPv6 packet SHOULD be copied from the entropy field in the BIER encapsulation.

2.1. IPv6 Options Considerations

For directly connected BIER routers, IPv6 Hop-by-Hop or Destination options are irrelevant and SHOULD NOT be inserted by BFIR on the BIERin6 packet. In this case IPv6 header, Next Header field should be set to TBD. Any IPv6 packet arriving on BFRs and BFERs, with multiple extension header where the last extension header has a Next Header field set to TBD, SHOULD be discard and the node should transmit an ICMP Parameter Problem message to the source of the packet (BFIR) with an ICMP code value of TBD10 ('invalid options for BIERin6').

This also indicates that for disjoint BIER routers using IPv6 encapsulation, there SHOULD NOT be any IPv6 Hop-by-Hop or Destination options be present in a BIERin6 packet. In this case, if additional traffic engineering is required, IPv6 tunneling (i.e. BIERin6 over SRv6) can be implemented.

3. BIER Header

The BIER header MUST be encoded per Section 2.2 of [RFC8296].

The BIFT-id is either encoded per [I-D.ietf-bier-non-mpls-bift-encoding] or per advertised by BFRs, as specified in [I-D.ietf-bier-lsr-ethernet-extensions].

4. IPv6 Encapsulation Advertisement

When IPv6 encapsulation is not required between directly connected BFRs, no signaling in addition to that specified in [I-D.ietf-bier-lsr-ethernet-extensions] is needed.

Otherwise, a node that requires IPv6 encapsulation MUST advertise the BIER IPv6 transportation sub-sub-sub-TLV/sub-sub-TLV according to local configuration or policy in the BIER domain to request other BFRs to always use IPv6 encapsulation.

In presence of multiple encapsulation possibilities hop-by-hop it is a matter of local policy which encapsulation is imposed and the receiving router MUST accept all encapsulations that it advertised.

4.1. Format

The BIER IPv6 transportation is a new sub-sub-TLV of BIER Ethernet Encapsulation sub-TLV defined in OSPFv3, and a new sub-sub-sub-TLV of BIER Ethernet Encapsulation sub-sub-TLV defined in ISIS, as per [I-D.ietf-bier-lsr-ethernet-extensions].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type       |   Length      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Type: For OSPF, value TBD1 (prefer 1) is used to indicate it is the IPv6 transportation sub-TLV. For ISIS, value TBD2 (prefer 1) is used to indicate it is the IPv6 transportation sub-sub-TLV.

- Length: 0.

4.2. Inter-area prefix redistribution

When BFR-prefixes are advertised across IGP areas per [I-D.ietf-bier-lsr-ethernet-extensions] or redistributed across protocol boundaries per [I-D.ietf-bier-prefix-redistribute], the BIER IPv6 transportation sub-sub-TLV or sub-sub-sub-TLV MAY be re-advertised/re-distributed as well.

5. IANA Considerations

IANA is requested to assign a new "BIER" type for "Next Header" in the "Assigned Internet Protocol Numbers" registry.

IANA is requested to assign a new "BIERin6" type for "invalid options" in the "ICMP code value" registry.

IANA is requested to assign a new "BIER IPv6 transportation Sub-sub-TLV" type in the "OSPFv3 BIER Ethernet Encapsulation sub-TLV" Registry.

IANA is requested to set up a new "BIER IPv6 transportation Sub-sub-sub-TLV" type in the "IS-IS BIER Ethernet Encapsulation sub-sub-TLV" Registry.

6. Security Considerations

General IPv6 and BIER security considerations apply.

7. Acknowledgement

The authors would like to thank Tony Przygienda, Nagendra Kumar for their review and valuable comments.

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, <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <https://www.rfc-editor.org/info/rfc8279>.

[RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <https://www.rfc-editor.org/info/rfc8296>.

[RFC8401] Ginsberg, L., Ed., Przygienda, T., Aldrin, S., and Z. Zhang, "Bit Index Explicit Replication (BIER) Support via IS-IS", RFC 8401, DOI 10.17487/RFC8401, June 2018, <https://www.rfc-editor.org/info/rfc8401>.

8.2. Informative References

[I-D.ietf-bier-bar-ipa]
Zhang, Z., Przygienda, T., Dolganow, A., Bidgoli, H., Wijnands, I., and A. Gulko, "BIER Underlay Path Calculation Algorithm and Constraints", draft-ietf-bier-bar-ipa-07 (work in progress), September 2020.

[I-D.ietf-bier-bfd]
Xiong, Q., Mirsky, G., hu, f., and C. Liu, "BIER BFD", draft-ietf-bier-bfd-00 (work in progress), November 2020.

[I-D.ietf-bier-idr-extensions]
Xu, X., Chen, M., Patel, K., Wijnands, I., and T. Przygienda, "BGP Extensions for BIER", draft-ietf-bier-idr-extensions-07 (work in progress), September 2019.

[I-D.ietf-bier-ipv6-requirements]
McBride, M., Xie, J., Geng, X., Dhanaraj, S., Asati, R., Zhu, Y., Mishra, G., and Z. Zhang, "BIER IPv6 Requirements", draft-ietf-bier-ipv6-requirements-09 (work in progress), September 2020.

[I-D.ietf-bier-lsr-ethernet-extensions]
Dhanaraj, S., Yan, G., Wijnands, I., Psenak, P., Zhang, Z., and J. Xie, "LSR Extensions for BIER over Ethernet", draft-ietf-bier-lsr-ethernet-extensions-02 (work in progress), December 2020.

[I-D.ietf-bier-non-mpls-bift-encoding]
Wijnands, I., Mishra, M., Xu, X., and H. Bidgoli, "An Optional Encoding of the BIFT-id Field in the non-MPLS BIER Encapsulation", draft-ietf-bier-non-mpls-bift-encoding-03 (work in progress), November 2020.

[I-D.ietf-bier-ospfv3-extensions]
Psenak, P., Nainar, N., and I. Wijnands, "OSPFv3 Extensions for BIER", draft-ietf-bier-ospfv3-extensions-03 (work in progress), November 2020.

[I-D.ietf-bier-path-mtu-discovery]
Mirsky, G., Przygienda, T., and A. Dolganow, "Path Maximum Transmission Unit Discovery (PMTUD) for Bit Index Explicit Replication (BIER) Layer", draft-ietf-bier-path-mtu-discovery-09 (work in progress), November 2020.

[I-D.ietf-bier-ping]
Nainar, N., Pignataro, C., Akiya, N., Zheng, L., Chen, M., and G. Mirsky, "BIER Ping and Trace", draft-ietf-bier-ping-07 (work in progress), May 2020.

[I-D.ietf-bier-pmmm-oam]
Mirsky, G., Zheng, L., Chen, M., and G. Fioccola, "Performance Measurement (PM) with Marking Method in Bit Index Explicit Replication (BIER) Layer", draft-ietf-bier-pmmm-oam-09 (work in progress), December 2020.

[I-D.ietf-bier-prefix-redistribute]
Zhang, Z., Wu, B., Zhang, Z., Wijnands, I., and Y. Liu, "BIER Prefix Redistribute", draft-ietf-bier-prefix-redistribute-00 (work in progress), August 2020.

[I-D.xzlnp-bier-ioam]
Min, X., Zhang, Z., Liu, Y., Nainar, N., and C. Pignataro, "Bit Index Explicit Replication (BIER) Encapsulation for In-situ OAM (IOAM) Data", draft-xzlnp-bier-ioam-00 (work in progress), July 2020.

[I-D.zhang-bier-babel-extensions]
Zhang, Z. and T. Przygienda, "BIER in BABEL", draft-zhang-bier-babel-extensions-04 (work in progress), November 2020.

[I-D.zzhang-tsvwg-generic-transport-functions]
Zhang, Z., Bonica, R., and K. Kompella, "Generic Transport Functions", draft-zzhang-tsvwg-generic-transport-functions-00 (work in progress), November 2020.

[RFC7368] Chown, T., Ed., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", RFC 7368, DOI 10.17487/RFC7368, October 2014, <https://www.rfc-editor.org/info/rfc7368>.

Authors' Addresses

Zheng(Sandy) Zhang
ZTE Corporation

EMail: zhang.zheng@zte.com.cn

Zhaohui Zhang (editor)
Juniper Networks

EMail: zzhang@juniper.net

IJsbrand Wijnands
Individual

EMail: ice@braindump.be

Mankamana Mishra
Cisco Systems

EMail: mankamis@cisco.com

Hooman Bidgoli
Nokia

EMail: hooman.bidgoli@nokia.com

Gyan Mishra (editor)
Verizon

EMail: gyan.s.mishra@verizon.com