Abstract

Virtual Private Networks (VPNs) provide different customers with logically separated connectivity over a common network infrastructure. With the introduction and evolution of 5G and other network scenarios, some existing or new customers may require connectivity services with advanced characteristics compared to traditional VPNs. Such kind of network service is called enhanced VPNs (VPN+).

A Virtual Transport Network (VTN) is a virtual underlay network which consists of a set of dedicated or shared network resources allocated from the physical underlay network, and is associated with a customized logical network topology. VPN+ services can be delivered by mapping one or a group of overlay VPNs to the appropriate VTNs as the virtual underlay. In packet forwarding, some fields in the data packet need to be used to identify the VTN the packet belongs to, so that the VTN-specific processing can be performed on each node the packet traverses.

This document proposes a new Hop-by-Hop option of IPv6 extension header to carry the VTN Resource ID, which is used to identify the set of network resources allocated to a VTN for packet processing. The procedure for processing the VTN option is also specified.
1. Introduction

Virtual Private Networks (VPNs) provide different customers with logically isolated connectivity over a common network infrastructure. With the introduction and evolvement of 5G and other network
scenarios, some existing or new customers may require connectivity services with advanced characteristics comparing to traditional VPNs, such as resource isolation from other services or guaranteed performance. Such kind of network service is called enhanced VPN (VPN+). VPN+ service requires the coordination and integration between the overlay VPNs and the network characteristics of the underlay.

[I-D.ietf-teas-enhanced-vpn] describes a framework and the candidate component technologies for providing VPN+ services. It also introduces the concept of Virtual Transport Network (VTN). A Virtual Transport Network (VTN) is a virtual underlay network which consists of a set of dedicated or shared network resources allocated from the physical underlay network, and is associated with a customized logical network topology. VPN+ services can be delivered by mapping one or a group of overlay VPNs to the appropriate VTNs as the underlay, so as to provide the network characteristics required by the customers. In packet forwarding, traffic of different VPN+ services need to be processed separately based on the network resources and the logical topology associated with the corresponding VTN.

[I-D.dong-teas-enhanced-vpn-vtn-scalability] describes the scalability considerations and the possible optimizations for providing a relatively large number of VTNs for VPN+ services. One approach to improve the data plane scalability of VTN is to introduce a dedicated VTN Resource Identifier (VTN Resource ID) in the data packet to identify the set of network resources allocated to a VTN, so that VTN-specific packet processing can be performed using that set of resources, which avoids the possible resource competition with services in other VTNs. This is called Resource Independent (RI) VTN. A VTN Resource ID represents a subset of the resources (e.g. bandwidth, buffer and queuing resources) allocated on a given set of links and nodes which constitute a logical network topology. The logical topology associated with a VTN could be defined using mechanisms such as Multi-Topology [RFC4915], [RFC5120] or Flex-Algo [I-D.ietf-lsr-flex-algo], etc.

This document proposes a mechanism to carry the VTN resource ID in a new Hop-by-Hop option of IPv6 extension header [RFC8200] of IPv6 packet, so that on each network node along the packet forwarding path, the VTN option in the packet is parsed, and the obtained VTN Resource ID is used to instruct the network node to use the set of network resources allocated to the corresponding VTN to process and forward the packet. The procedure for processing the VTN Resource ID is also specified. This provides a scalable solution to support a relatively large number of VTNs in an IPv6 network.
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 BCP14 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. New IPv6 Extension Header Option for VTN

A new Hop-by-Hop option type "VTN" is defined to carry the VTN related Identifier in an IPv6 packet. Its format is shown as below:

Option   Option       Option
Type     Data Len     Data
+--------+--------+-------------------------+
|BBCTTTTT|00000100| 4-octet VTN Resource 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:

- 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.

- 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.

VTN Resource ID: 4-octet identifier which uniquely identifies the set of network resources allocated to a VTN.

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

```
+------------+-------------------------+
|    SST     |   Slice Differentiator  |
+------------+-------------------------+
```

Figure 2. The format of S-NSSAI
3. 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 either processed or ignored in forwarding plane by all the nodes along the path.

3.1. VTN Option Insertion

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

3.2. VTN based Packet Forwarding

On receipt of a packet with the VTN option, each network node which can process the VTN option in fast path SHOULD use the VTN Resource ID to determine the set of local network resources allocated to the VTN for packet processing. The packet forwarding behavior is based on both the destination IP address and the VTN Resource ID. More specifically, the destination IP address is used to determine the next-hop and the outgoing interface, and VTN Resource ID is used to determine the set of network resources on the outgoing interface which are reserved to the VTN for processing and sending the packet. The Traffic Class field of the outer IPv6 header MAY be used to provide Diffserv treatment for packets which belong to the same VTN. The egress node of the IPv6 domain SHOULD decapsulate the outer IPv6 header which includes the VTN option.

In the forwarding plane, there can be different approaches of partitioning the local network resources and allocating them to different VTNs. For example, on one physical interface, a subset of the forwarding plane resources (e.g. the bandwidth and the associated buffer and queuing resources) can be allocated to a particular VTN and represented as a virtual sub-interface with reserved bandwidth resource. In packet forwarding, the IPv6 destination address of the received packet is used to identify the next-hop and the outgoing layer-3 interface, and the VTN Resource ID is used to further identify the virtual sub-interface which is associated with the VTN on the outgoing interface.

Network nodes which do not support the processing of Hop-by-Hop options header SHOULD ignore the Hop-by-Hop options header and
forward the packet only based on the destination IP address. Network nodes which support Hop-by-Hop Options header, but do not support the VTN option SHOULD ignore the VTN option and continue to forward the packet based on the destination IP address and MAY also based on the rest of the Hop-by-Hop Options.

4. Operational Considerations

As described in [RFC8200], network 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. The proposed modification to the processing of IPv6 Hop-by-Hop options header is specified in [I-D.hinden-6man-hbh-processing]. Operator needs to make sure that all the network nodes involved in a VTN can either process Hop-by-Hop Options header in the fast path, or ignore the Hop-by-Hop Option header. Since a VTN is associated with a logical network topology, it is practical to ensure that all the network nodes involved in that logical topology support the processing of the HBH options header and the VTN option. 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 network nodes involved in a VTN to process the Hop-by-Hop Options header and the VTN option if there is a nob for this.

5. IANA Considerations

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>VTN Option</td>
<td>this document</td>
</tr>
</tbody>
</table>

6. Security Considerations

The security considerations with IPv6 Hop-by-Hop options header are described in [RFC8200], [RFC7045] and [I-D.hinden-6man-hbh-processing]. This document introduces a new IPv6 Hop-by-Hop option which is either processed in the fast path or ignored by network nodes, thus it does not introduce additional security issues.

7. Contributors
8. Acknowledgements

The authors would like to thank Juhua Xu, James Guichard, Joel Halpern and Tom Petch for their review and valuable comments.

9. References

9.1. Normative References

[I-D.ietf-teas-enhanced-vpn]


9.2. Informative References

[I-D.dong-teas-enhanced-vpn-vtn-scalability]

[I-D.hinden-6man-hbh-processing]
[I-D.ietf-lsr-flex-algo]


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

Gyan Mishra
Verizon Inc.

Email: gyan.s.mishra@verizon.com
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

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

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

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

This Internet-Draft will expire on May 2, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Herbert                    Expires May 2, 2021
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|  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 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
          | Type                  | Local_ID |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
          | E |  Num opts              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

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.
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 |------+
+++ +++
| Inserted options |
+++++++
| Attribution Opt |
+++ +++
| 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:

$$7 - ((\text{offset\_last\_inserted\_byte} - 3) \mod 8)$$

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:

\[
7 - \left( (\text{offset}\_\text{last}\_\text{inserted}\_\text{byte} - 2) \mod 8 \right)
\]

where \(\text{offset}\_\text{last}\_\text{byte}\) 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 assign the following Destination and Hop-By-Hop option:

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Binary value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>00 0 TBD</td>
<td>Attribution Option</td>
<td>This document</td>
</tr>
</tbody>
</table>

Herbert                      Expires May 2, 2021
6. References

6.1. Normative References


6.2. Informative References


Internet-Draft Attribution Option October 2020


Author’s Address

Tom Herbert
Intel
Santa Clara, CA
USA

Email: tom@quantonium.net
IPv6 Maintenance J. Linkova
Internet-Draft Google
Updates: 4861 (if approved) July 5, 2021
Intended status: Standards Track
Expires: January 6, 2022

Gratuitous Neighbor Discovery: Creating Neighbor Cache Entries on First-Hop Routers
draft-ietf-6man-grand-07

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 initiates connections to an off-link destination from a new IPv6 address.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 6, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

1.  Introduction .................................................. 3
  1.1.  Requirements Language ................................... 3
  1.2.  Terminology ................................................. 4
2.  Problem Statement ............................................. 4
3.  Solution Requirements ......................................... 6
4.  Changes to Neighbor Discovery ................................. 6
  4.1.  Nodes Sending Gratuitous Neighbor Advertisements .......... 7
  4.2.  Routers Creating Cache Entries Upon Receiving Unsolicited
        Neighbor Advertisements .................................... 7
5.  Avoiding Disruption ............................................ 8
  5.1.  Neighbor Cache Entry Exists in Any State Other Than
        INCOMPLETE ................................................. 9
  5.2.  Neighbor Cache Entry is in INCOMPLETE state ............... 9
  5.3.  Neighbor Cache Entry Does Not Exist ........................ 10
        5.3.1.  The Rightful Owner Is Not Sending Packets From The
                Address ............................................. 11
        5.3.2.  The Rightful Owner Has Started Sending Packets From
                The Address ......................................... 12
6.  Modifications to RFC-Mandated Behavior ....................... 13
  6.1.  Modification to RFC4861 Neighbor Discovery for IP version
        6 (IPv6) .................................................. 13
        6.1.1.  Modification to the section 7.2.5 .................... 13
        6.1.2.  Modification to the section 7.2.6 .................... 14
7.  Solution Limitations ........................................... 15
8.  Solutions Considered but Discarded ........................... 16
  8.1.  Do Nothing ................................................ 16
  8.2.  Change to the Registration-Based Neighbor Discovery ....... 16
  8.3.  Host Sending NS to the Router Address from Its GUA ....... 17
  8.4.  Host Sending Router Solicitation from its GUA ............. 17
  8.5.  Routers Populating Their Caches by Gleaning From Neighbor
        Discovery Packets ......................................... 18
  8.6.  Initiating Hosts-to-Routers Communication .................. 18
  8.7.  Making the Probing Logic on Hosts More Robust ............. 19
  8.8.  Increasing the Buffer Size on Routers ..................... 20
  8.9.  Transit Dataplane Traffic From a New Address Triggering
        Address Resolution ......................................... 20
9.  IANA Considerations ........................................... 20
10. Security Considerations ........................................ 21
11. Acknowledgements ............................................. 22
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 containing A’s IPv6 and link-layer addresses 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]. Any additional packets arriving before the resolution process finishes are likely to result in dropped packets. It can cause packet loss and performance degradation that can be user-visible.

This document updates the Neighbor Discovery protocol [RFC4861] to avoid packet loss in the scenario described above. Section 4 discusses the changes and analyses the potential impact, while normative changes to [RFC4861] are specified in Section 6.

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
1.2. Terminology

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

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

ND: Neighbor Discovery, [RFC4861].

NC: Neighbor Cache, [RFC4861]. The Neighbor Cache entry can be in one of five states, as described in section 7.3.2 of [RFC4861]: INCOMPLETE, REACHABLE, STALE, DELAY, PROBE.

SLAAC: IPv6 Stateless Address Autoconfiguration, [RFC4862].

NS: Neighbor Solicitation, [RFC4861].

NA: Neighbor Advertisement, [RFC4861].

RS: Router Solicitation, [RFC4861].

RA: Router Advertisement, [RFC4861].

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

TLLAO: Target link-layer Address Option, 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].

Preferred Address: an address assigned to an interface whose uniqueness has been verified using DAD and whose use by upper-layer protocols is unrestricted, [RFC4862]. Preferred addresses may be used as the source address of packets sent from the interface.

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 sent from the router’s link-local address to a link-local destination address 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 and 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 any of the device’s 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 (Section 7.3.2 of [RFC4861]). 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, and some implementations are known to buffer just one packet. So any subsequent packets arriving before the address resolution process is completed are causing packet loss by replacing older packets in the buffer.

4. If the host sends multiple probes in parallel, in the worst case, 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 [RFC8981] 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:

- If the router does not have a Neighbor Cache entry for the address, a STALE entry needs to be created proactively, prior to arrival of the first packet intended for that address.

- 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.

- In case of duplicate addresses present in the network, the proposed solution should not override the existing entry.

- 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 needs to be compatible with the recommendations provided in [RFC6583].

4. Changes to Neighbor Discovery

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

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

- A router creates a new cache entry upon receiving an unsolicited NA from a host.
The following sections discuss these changes in more detail. Normative changes are specified in Section 6.

4.1. Nodes Sending Gratuitous Neighbor Advertisements

The section 7.2.6 of [RFC4861] discusses using unsolicited Neighbor Advertisements 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 the 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 routers on the given network segment/link, instead of being flooded to all nodes.

It should be noted that the proposed mechanism does not cause any significant increase in 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 this procedure would only produce an increase in the overall amount of multicast traffic if no return traffic arrives for the address that sent the unsolicited NA or if the router does not create a STALE entry upon receiving such NA. The increase would be negligible as that additional traffic is a few orders of magnitude less than the usual level of Neighbor Discovery multicast traffic.

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 to create 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 for an address that does not already have a Neighbor Cache entry. The proposed changes do not modify routers behaviour specified in [RFC4861] for the scenario when the corresponding Neighbor Cache entry already exists.

The next section analyses various scenarios of duplicated addresses and discusses the potential impact of creating a STALE entry for a duplicated IPv6 address.

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. 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 discuss the address collision scenario when a node sends an unsolicited NA for an address in the Optimistic state, while another node (the rightful owner) has the same address assigned already. This document uses the term "the rightful owner" as the same terminology is used in [RFC4429]. The analysis assumes that the host performs Duplicate Address Detection, as section 5.4 of [RFC4862] requires that DAD MUST be performed on all unicast addresses prior to assigning them to an interface.
5.1. Neighbor Cache Entry Exists in Any State Other Than 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 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.

The return traffic intended for the host with the Optimistic address would be sent to the rightful owner. However, this is unavoidable with or without the unsolicited NA mechanism.

5.2. Neighbor Cache Entry is in INCOMPLETE state

Another corner case is the INCOMPLETE cache entry for the address.

1. The router receives a packet for the rightful owner of the address.

2. The router starts the address resolution process by creating an INCOMPLETE entry and sends the multicast NS.

3. More packets arrive at the router for the address in question.

4. The host configures an Optimistic address and sends an unsolicited NA.

5. The router creates a STALE entry and sends the buffered packet(s) to the host (while at least some of those packets are actually intended for the rightful owner).

6. As the STALE entry was used to send packets, the router changes the entry state to DELAY and waits up to DELAY_FIRST_PROBE_TIME ([RFC4861], 5 secs) before sending unicast NS.

7. The rightful owner responds to the multicast NS sent at Step 2 with a solicited NA with the Override flag set.

8. The router updates the entry with the TLLAO supplied (the rightful owner link-layer address) and sets the entry state to REACHABLE (as the NA has the Solicited flag set).

As a result some packets (ones in the buffer at Step 6 and all packets arriving between Step 6 and Step 8) are delivered to the host with the Optimistic address, while some of them, if not all, are
intended for the rightful owner. Without the unsolicited NA, packet which are in the buffer at Step 8 (usually just one packet but some routers may buffer a few) would have been delivered to the rightful owner and the rest of the packets would have been dropped. However, the probability of such scenario is rather low as it would require the following things to happen almost simultaneously (within tens of milliseconds in most cases):

- One host starts using a new IPv6 address and sending traffic without sending an unsolicited NA first.
- Another host configures the same IPv6 address in Optimistic mode before the router completes the address resolution for the rightful owner.

It should be noted that in this scenario the rightful owner does not send any unsolicited NAs before sending packets. If the rightful owner implements the functionality described in this document and sends unsolicited NAs upon configuring its address, then the router creates a STALE entry for the address, causing all packets are delivered to the rightful owner (see Section 5.1). The rightful owner would experience no disruption but might receive some packets intended for the host with Optimistic address.

This section focuses on the scenario when the solicited NA from the rightful owner arrives after the unsolicited one sent from the Optimistic address (Step 7 and Step 4 respectively). If the solicited NA arrives first it changes the NC entry state from INCOMPLETE to REACHABLE. As discussed in Section 5.1, there will be no disruption for the rightful owner if the router already has a REACHABLE entry for the address when an unsolicited NA is received.

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 off-link communication recently or has never used it at all.
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 STAILE 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 STAILE it sends traffic using that entry, changes it to DELAY and waits 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. (It is unlikely that the host has not completed the DAD process at this stage, as DELAY_FIRST_PROBE_TIME (5 seconds) is much higher than the DAD duration (DupAddrDetectTransmits*RetransTimer*1000 + MAX_RTR_SOLICITATION_DELAY secs, section 5.4 of [RFC4862]). The default value for the DAD process would be 1*1*1000 + 1 = 2 secs, [RFC4861]). If the host has completed DAD but did not detect the address conflict then there are two hosts with the same address in the Preferred state and the disruption is inevitable anyway.

8. As the router receives no response for the unicast NSes, it deletes the NC entry.

9. If return packets for communication initiated at step 2 are still arriving, the router buffers a small number of those packets and starts the address resolution again by sending 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 buffered packet(s) are sent to that address. Any packets still arriving after the address resolution still completed are sent to the rightful address owner as well.
The rightful owner is not experiencing any disruption as it does not send any traffic. It would only start receiving packets intended for another host after Step 8 is completed and only if return packets for the communication initiated at step 2 are still arriving.

However, the same behaviour would be observed if changes proposed in this document are not implemented. If the host starts sending packets from its Optimistic address but then changes the address state to Duplicated, the first return packet would trigger the address resolution process and would be buffered until the resolution is completed. The buffered packet(s) and any packets still arriving after the address is resolved would be forwarded to the rightful owner of the address. So the rightful owner might still receive one or more packets from the flows intended for another host. Therefore, it’s safe to conclude that the proposed changes do introduce any disruption for the rightful owner of the duplicated address.

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 for the IPv6 address in question. Some flows intended for the rightful owner of the duplicated address, while some are for the new host. As the NC entry is STALE it sends traffic using that entry, changes it to DELAY and waits 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. As the router receives no response for the unicast NSes, it deletes the NC entry.

10. The next packet re-creates the entry and triggers the resolution process. The router buffers the packet and 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 \( \text{DELAY_FIRST_PROBE_TIME} \times 1000 + (\text{MAX_UNICAST_SOLICIT} - 1) \times \text{RetransTimer} \) milliseconds. As per the constants defined in Section 10 of [RFC4861] this interval is equal to \( 5 \times 1000 + (3 - 1) \times 1000 = 7000 \text{ms} \) or 7 seconds.

However, it should be noted that the probability of such scenario is rather low. Similar to the scenario discussed in Section 5.2, it would require the following things to happen almost simultaneously (within tens of milliseconds in most cases):

- One host starts using a new IPv6 address and sending traffic without sending an unsolicited NA first.
- Another host configures the same IPv6 address in Optimistic mode before the router receives the return traffic for the first host.

As discussed in Section 5.2, the disruption to the rightful owner can easily be prevented if that node implements the mechanism described in the document. Sending unsolicited NAs before initiating off-link communication would create a STALE entry in the router NC and prevent any traffic to that address to be sent to the host with the Optimistic address (see Section 5.1).

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 makes the following changes to the section 7.2.5 of [RFC4861]:
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.

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’s reachability state MUST 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]:

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:

- if Optimistic DAD [RFC4429] is used: a new Optimistic address is assigned to the node interface.
- 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 starting 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:

- 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.
- 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:

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

- Unhappy users.
- Many support tickets.

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, significant changes to the existing IPv6 implementations would be needed, so
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:

- The NS source address is the host GUA.
- The destination address is the default router IPv6 address.
- The Source Link-Layer Address option contains the host link-layer address.
- 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:

- Would not work for Optimistic addresses as section 2.2 of [RFC4429] explicitly prohibits sending Neighbor Solicitations from an Optimistic Address.
- 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.
- 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 cannot 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 cannot 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
This approach has the following drawbacks:

- Data packets to the router link-local address could be blocked by security policy or control plane protection mechanism.
- It introduces an additional overhead for routers control plane (in addition to processing ND packets, the data packet needs to be processed as well).
- 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:

- It would require updating all possible applications performing probing, while the proposed solution is implemented on operating systems level.
- Some implementations need to send multiple probes. Examples include but not limited to:
  - Sending AAAA and A records DNS probes in parallel.
  - Detecting captive portals often require sending multiple packets.
- 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 experience the issue.

o Reducing the probe retransmit interval unnecessarily increases the network utilization and might cause the network congestion.

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 sourced by an on-link neighbor node, 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. The proactive address resolution must only be triggered by packets from the prefixes known to be on-link. Otherwise, traffic from spoofed source addresses or any transit traffic could lead to neighbor cache exhaustion.

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, there are no new vectors for an attacker to override an existing cache entry.

Section 5 describes some corner cases when a host with the duplicated Optimistic address might get some packets intended for the rightful owner of the address. However such scenarios do not introduce any new attack vectors: even without the proposed changes, an attacker can easily override the routers neighbor cache and redirect the traffic by sending NAs with the Solicited flag set. As discussed in Section 5.3.2 the worst case scenario might cause a disruption for up to 7 seconds. This risk is considered acceptable due to very low probability of that scenario. More importantly, for all cases described in Section 5 the rightful owner can prevent disruption caused by an accidental address duplication just by implementing the mechanism described in this document. If the rightful owner sends unsolicited NAs before using the address, the STALE entry would be created on the router NC and any subsequent unsolicited NAs sent from the host with an Optimistic address would not override the NC 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 if MLD snooping is not enabled. 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 layer-2 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, Roman Danyliw, Owen DeLong, Martin Duke, Igor Gashinsky, Carles Gomez, Fernando Gont, Tatuya Jinmei, Benjamin Kaduk, Scott Kelly, Erik Kline, Warren Kumari, Barry Leiba, Jordi Palet Martinez, Erik Nordmark, Michael Richardson, Dan Romascun, Zaheduzzaman Sarker, Michael Scharf, John Scudder, Mark Smith, Dave Thaler, Pascal Thubert, Loganaden Velvindron, Eric Vyncke.

12. References

12.1. Normative References


12.2. Informative References

[I-D.halpern-6man-nd-pre-resolve-addr]


Author’s Address

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

Email: furry@google.com
IPv6 Application of the Alternate Marking Method

draft-ietf-6man-ipv6-alt-mark-14

Abstract

This document describes how the Alternate Marking Method can be used as a passive performance measurement tool in an IPv6 domain. It defines a new Extension Header Option to encode Alternate Marking information in both the Hop-by-Hop Options Header and Destination Options Header.

Status of This Memo

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

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

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

This Internet-Draft will expire on October 30, 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

[I-D.ietf-ippm-rfc8321bis] and [I-D.ietf-ippm-rfc8889bis] 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 to as the Alternate Marking Method.

This document defines how the Alternate Marking Method can be used to measure performance metrics in IPv6. The rationale is to apply the Alternate Marking methodology to IPv6 and therefore allow detailed packet loss, delay and delay variation measurements both hop-by-hop and end-to-end to exactly locate the issues in an IPv6 network.

The Alternate Marking is an on-path telemetry technique and consists of synchronizing the measurements in different points of a network by
switching the value of a marking bit and therefore dividing the packet flow into batches. Each batch represents a measurable entity recognizable by all network nodes along the path. By counting the number of packets in each batch and comparing the values measured by different nodes, it is possible to precisely measure the packet loss. Similarly, the alternation of the values of the marking bits can be used as a time reference to calculate the delay and delay variation. The Alternate Marking operation is further described in Section 5.

The format of IPv6 addresses is defined in [RFC4291] while [RFC8200] defines the IPv6 Header, including a 20-bit Flow Label and the IPv6 Extension Headers.

This document introduces a new TLV (type-length-value) that can be encoded in the Options Headers (Hop-by-Hop or Destination) for the purpose of the Alternate Marking Method application in an IPv6 domain.

The threat model for the application of the Alternate Marking Method in an IPv6 domain is reported in Section 6. As with all on-path telemetry techniques, the only definitive solution is that this methodology MUST be applied in a controlled domain.

1.1. Terminology

This document uses the terms related to the Alternate Marking Method as defined in [I-D.ietf-ippm-rfc8321bis] and [I-D.ietf-ippm-rfc8889bis].

1.2. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Alternate Marking application to IPv6

The Alternate Marking Method requires a marking field. Several alternatives could be considered such as IPv6 Extension Headers, IPv6 Address and Flow Label. But, it is necessary to analyze the drawbacks for all the available possibilities, more specifically:

Reusing existing Extension Header for Alternate Marking leads to a non-optimized implementation;
Using the IPv6 destination address to encode the Alternate Marking processing is very expensive;

Using the IPv6 Flow Label for Alternate Marking conflicts with the utilization of the Flow Label for load distribution purpose ([RFC6438]).

In the end, a new Hop-by-Hop or a new Destination Option is the best choice.

The approach for the Alternate Marking application to IPv6 specified in this memo is compliant with [RFC8200]. It 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). The intermediate nodes and destination node MUST only read the marking values of the option without modifying the Option Header.

- 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 and the measurement can be done only for the nodes configured to read the Option. As further discussed in Section 4, the presence of the hop-by-hop option should not affect the traffic throughput both on nodes that do not recognize this option and on the nodes that support it. However, it is worth mentioning that there is a difference between theory and practice. Indeed, in a real implementation it can happen that packets with hop-by-hop option could also be skipped or processed in the slow path. While some proposals are trying to address this problem and make Hop-by-Hop Options more practical ([I-D.peng-v6ops-hbh], [I-D.hinden-6man-hbh-processing]), these aspects are out of the scope for this document.

- 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. However, as said, routers will only examine this option if properly 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 Section 5.3, goes in this direction and it is used to identify a monitored flow.

The FlowMonID is different from the Flow Label field of the IPv6 Header ([RFC6437]). The Flow Label field in the IPv6 header is used by a source to label sequences of packets to be treated in the network as a single flow and, as reported in [RFC6438], it can be used for load-balancing/equal cost multi-path (LB/ECMP). The reuse of Flow Label field for identifying monitored flows is not considered because it may change the application intent and forwarding behavior. Also, the Flow Label may be changed en route and this may also invalidate the integrity of the measurement. Furthermore, since the Flow Label is pseudo-random, there is always a finite probability of collision. Those reasons make the definition of the FlowMonID necessary for IPv6. Indeed, the FlowMonID is designed and only used to identify the monitored flow. Flow Label and FlowMonID within the same packet are totally disjoint, have different scope, are used to identify flows based on different criteria, and are intended for different use cases.

The rationale for the FlowMonID is further discussed in Section 5.3. This 20 bit field allows easy and flexible identification of the monitored flow and enables improved measurement correlation and finer granularity since it can be used in combination with the traditional TCP/IP 5-tuple to identify a flow. An important point that will be discussed in Section 5.3 is the uniqueness of the FlowMonID and how to allow disambiguation of the FlowMonID in case of collision.

The following section highlights an important requirement for the application of the Alternate Marking to IPv6. The concept of the controlled domain is explained and it is considered an essential precondition, as also highlighted in Section 6.

2.1. Controlled Domain

[RFC8799] introduces the concept of specific limited domain solutions and, in this regard, it is reported the IPv6 Application of the Alternate Marking Method as an example.

IPv6 has much more flexibility than IPv4 and innovative applications have been proposed, but for a number of reasons, such as the policies, options supported, the style of network management and security requirements, it is suggested to limit some of these applications to a controlled domain. This is also the case of the Alternate Marking application to IPv6 as assumed hereinafter.
Therefore, the IPv6 application of the Alternate Marking Method MUST be deployed in a controlled domain. It is RECOMMENDED that an implementation filters packets that carry Alternate Marking data and are entering or leaving the controlled domains.

A controlled domain is a managed network where it is required to select, monitor and control the access to the network by enforcing policies at the domain boundaries in order to discard undesired external packets entering the domain and check the internal packets leaving the domain. It does not necessarily mean that a controlled domain is a single administrative domain or a single organization. A controlled domain can correspond to a single administrative domain or can be composed by multiple administrative domains under a defined network management. Indeed, some scenarios may imply that the Alternate Marking Method involves more than one domain, but in these cases, it is RECOMMENDED that the multiple domains create a whole controlled domain while traversing the external domain by employing IPsec [RFC4301] authentication and encryption or other VPN technology that provides full packet confidentiality and integrity protection.

In a few words, it must be possible to control the domain boundaries and eventually use specific precautions if the traffic traverse the Internet.

The security considerations reported in Section 6 also highlight this requirement.

2.1.1. Alternate Marking Measurement Domain

The Alternate Marking measurement domain can overlap with the controlled domain or may be a subset of the controlled domain. The typical scenarios for the application of the Alternate Marking Method depend on the controlled domain boundaries, in particular:

the user equipment can be the starting or ending node, only in case it is fully managed and if it belongs to the controlled domain. In this case the user generated IPv6 packets contain the Alternate Marking data. But, in practice, this is not common due to the fact that the user equipment cannot be totally secured in the majority of cases.

the CPE (Customer Premises Equipment) is most likely to be the starting or ending node since it connects the user’s premises with the service provider’s network and therefore belongs to the operator’s controlled domain. Typically the CPE encapsulates a received packet in an outer IPv6 header which contains the Alternate Marking data. The CPE can also be able to filter and drop packets from outside of the domain with inconsistent fields to make effective the relevant security rules at the domain
boundaries, for example a simple security check can be to insert
the Alternate Marking data if and only if the destination is
within the controlled domain.

3. Definition of the AltMark Option

The definition of a new TLV for the Options Extension Headers,
carrying the data fields dedicated to the Alternate Marking method,
is reported below.

3.1. Data Fields Format

The following figure shows the data fields format for enhanced
Alternate Marking TLV (AltMark). This AltMark data can 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Option Type  |  Opt Data Len |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              FlowMonID                |L|D|     Reserved      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- **Option Type**: 8-bit identifier of the type of Option that needs to
  be allocated. Unrecognized Types MUST be ignored on processing.
  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 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). In this way, since the three high-order bits of the
  AltMark Option are set to 000, it means that nodes can simply skip
  this Option if they do not recognize and that the data of this
  Option do not change en route, indeed the source is the only one
  that can write it.

- **Opt Data Len**: 4. It is the length of the Option Data Fields of
  this Option in bytes.
FlowMonID: 20-bit unsigned integer. The FlowMon identifier is described in Section 5.3. As further discussed below, it has been picked as 20 bits since it is a reasonable value and a good compromise in relation to the chance of collision. It MUST be set pseudo randomly by the source node or by a centralized controller.

L: Loss flag for Packet Loss Measurement as described in Section 5.1;

D: Delay flag for Single Packet Delay Measurement as described in Section 5.2;

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 it is 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 processes 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.

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. In many cases the end-to-end measurement is not enough and it is required the hop-by-hop measurement, so the most complete choice can be 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.

Another scenario that can be mentioned is the presence of a Routing Header, in particular it is possible to consider SRv6. A new type of Routing Header, referred as Segment Routing Header (SRH), has been defined in [RFC8754] for SRv6. Like any other use case of IPv6, Hop-
by-Hop and Destination Options are usable 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 SR node that is identified by the SR path. More details about the SRv6 application are described in [I-D.fz-spring-srv6-alt-mark].

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

- Destination Option not preceding a Routing Header => measurement only by node in Destination Address.
- Hop-by-Hop Option => every router on the path with feature enabled.
- Destination Option preceding a 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 (Operations, Administration, and Maintenance). 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.

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. The definition of the Hop-by-Hop Options in this document is also designed to minimize throughput impact both on nodes that do not recognize the Option and on node that support it. Indeed, the three high-order bits of the Options Header defined in this draft are 000 and, in theory, as per [RFC8200] and [I-D.hinden-6man-hbh-processing], this means "skip if do not recognize and data do not change en route". [RFC8200] also mentions that the nodes only examine and process the Hop-by-Hop Options header.
if explicitly configured to do so. For these reasons, this Hop-by-Hop Option should not affect the throughput. However, in practice, it is important to be aware that the things may be different in the implementation and it can happen that packets with Hop-by-Hop are forced onto the slow path, but this is a general issue, as also explained in [I-D.hinden-6man-hbh-processing]. It is also worth mentioning that the application to a controlled domain should avoid the risk of arbitrary nodes dropping packets with Hop-by-Hop Options.

5. Alternate Marking Method Operation

This section describes how the method operates. [I-D.ietf-ippm-rfc8321bis] introduces several applicable methods which are reported below, and a new field is introduced to facilitate the deployment and improve the scalability.

5.1. Packet Loss Measurement

The measurement of the packet loss is really straightforward in comparison to the existing mechanisms, as detailed in [I-D.ietf-ippm-rfc8321bis]. 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. The source node is the only one that marks the packets to create the batches, while the intermediate nodes only read the marking values and identify the packet batches. 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 recognizable by all network nodes along the path.

Both fixed number of packets and fixed timer can be used by the source node to create packet batches. But, as also explained in [I-D.ietf-ippm-rfc8321bis], the timer-based batches are preferable because they are more deterministic than the counter-based batches. There is no definitive rule for counter-based batches, differently from timer-based batches. Using a fixed timer for the switching offers better control over the method, indeed the length of the batches can be chosen large enough to simplify the collection and the comparison of the measures taken by different network nodes. In the implementation the counters can be sent out by each node to the controller that is responsible for the calculation. It is also possible to exchange this information by using other on-path techniques. But this is out of scope for this document.
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. The mathematical formula on timing aspects, explained in section 5 of [I-D.ietf-ippm-rfc8321bis], must be satisfied and it takes into considerations the different causes of reordering such as clock error and network delay. The assumption is to define the available counting interval where to get stable counters and to avoid these issues. Specifically, if the effects of network delay are ignored, the condition to implement the methodology is that the clocks in different nodes MUST be synchronized to the same clock reference with an accuracy of +/- B/2 time units, where B is the fixed time duration of the batch, which refers to the original marking interval at the source node considering that this interval could fluctuate along the path. In this way each marked packet can be assigned to the right batch by each node. 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. The length of the batches can be determined based on the specific deployment scenario.

![Packet Loss Measurement and Single-Marking Methodology using L bit](image)

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

It is worth mentioning that the duration of the batches is considered stable over time in the previous figure. In theory, it is possible to change the length of batches over time and among different flows for more flexibility. But, in practice, it could complicate the correlation of the information.
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. But 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. Regarding the rule for the selection of the packet to be double-marked, the same considerations in Section 5.1 apply also here and the double-marked packet can be chosen within the available counting interval that is not affected by factors such as clock errors. 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.

In summary the approach with double marking is better than the approach with single marking. Moreover, the two approaches provide slightly different pieces of information and the data consumer can combine them to have a more robust data set.

Similar to what said in Section 5.1 for the packet counters, in the implementation the timestamps can be sent out to the controller that is responsible for the calculation or could also be exchanged using other on-path techniques. But this is out of scope for this document.

L bit=1        +-----------+           +-----------+           +----------
L bit=0        +-----------+           +-----------+           +----------
D bit=1        +          +          +          +            +
D bit=0        +-----------+----------+----------+------------+-----

Traffic Flow
===========================================================>
L bit   ...1111111111 0000000000 1111111111 0000000000 111111111...
D bit   ...0000010000 0000010000 0000010000 0000100000 000001000...
===========================================================>

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

Likewise 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) identifies the flow to be measured and 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, indeed, it can be used together with other identifiers (e.g. 5-tuple).
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 FlowMonID MUST only be used as a monitored flow identifier in order to determine a monitored flow within the measurement domain. This entails not only an easy identification but improved correlation as well.

The value of 20 bits has been selected for the FlowMonID since it is a good compromise and implies a low rate of ambiguous FlowMonIDs that can be considered acceptable in most of the applications. The disambiguation issue can be solved by tagging the pseudo randomly generated FlowMonID with additional flow information. In particular, it is RECOMMENDED to consider the 3-tuple FlowMonID, source and destination addresses:

- If the 20 bit FlowMonID is set independently and pseudo randomly in a distributed way there is a chance of collision. Indeed, by 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 1206 flows. So, for more entropy, FlowMonID is combined with source and destination addresses. Since there is a 1% chance of collision for 145 flows, it is possible to monitor 145 concurrent flows per host pairs with a 1% chance of collision.

- If the 20 bits FlowMonID is set pseudo randomly but in a centralized way, the controller can instruct the nodes properly in order to guarantee the uniqueness of the FlowMonID. With 20 bits, the number of combinations is 1048576, and the controller should ensure that all the FlowMonID values are used without any collision. Therefore, by considering source and destination addresses together with the FlowMonID, it can be possible to monitor 1048576 concurrent flows per host pairs.

A consistent approach MUST be used in the Alternate Marking deployment to avoid the mixture of different ways of identifying. All the nodes along the path and involved into the measurement SHOULD use the same mode for identification. As mentioned, it is RECOMMENDED to use the FlowMonID for identification purpose in combination with source and destination addresses to identify a flow. By considering source and destination addresses together with the FlowMonID it can be possible to monitor 145 concurrent flows per host pairs with a 1% chance of collision in case of pseudo randomly
generated FlowMonID, or 1048576 concurrent flows per host pairs in case of centralized controller. It is worth mentioning that the solution with the centralized control allows finer granularity and therefore adds even more flexibility to the flow identification.

The FlowMonID field is set at the source node, which is the ingress point of the measurement domain, and can be set in two ways:

a. It can be algorithmically generated by the source node, that can set it pseudo-randomly with some chance of collision. This approach cannot guarantee the uniqueness of FlowMonID since conflicts and collisions are possible. But, considering the recommendation to use FlowMonID with source and destination addresses the conflict probability is reduced due to the FlowMonID space available for each endpoint pair (i.e. 145 flows with 1% chance of collision).

b. It can be assigned by the central controller. Since the controller knows the network topology, it can allocate the value properly to avoid or minimize ambiguity and guarantee the uniqueness. In this regard, the controller can verify that there is no ambiguity between different pseudo-randomly generated FlowMonIDs on the same path. The conflict probability is really small given that the FlowMonID is coupled with source and destination addresses and up to 1048576 flows can be monitored for each endpoint pair. When all values in the FlowMonID space are consumed, the centralized controller can keep track and reassign the values that are not used any more by old flows.

If the FlowMonID is set by the source node, the intermediate nodes can read the FlowMonIDs from the packets in flight and act accordingly. While, if the FlowMonID is set by the controller, both possibilities are feasible for the intermediate nodes which can learn by reading the packets or can be instructed by the controller.

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 [I-D.ietf-ippm-rfc8889bis].

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.

There are documents on the control plane mechanisms of Alternate Marking, e.g. [I-D.ietf-idr-sr-policy-ifit], [I-D.chen-pce-pcep-ifit].

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. As already discussed in Section 4, it is RECOMMENDED that the AltMark Option does not affect the throughput and therefore the user experience.

Harm to the measurement: Alternate Marking measurements could be harmed by routers altering the fields of the AltMark Option (e.g. marking of the packets, FlowMonID) or by a malicious attacker adding AltMark Option to the packets in order to consume the resources of network devices and entities involved. As described above, the source node is the only one that writes the Option Header while the intermediate nodes and destination node only read it without modifying the Option Header. But, for example, an on-path attacker can modify the flags, whether intentionally or accidentally, or deliberately insert a new option to the packet flow or delete the option from the packet flow. The consequent effect could be to give the appearance of loss or delay or invalidate the measurement by modifying option identifiers, such as FlowMonID. The malicious implication can be to cause actions from the network administrator where an intervention is not necessary or to hide real issues in the
network. Since the measurement itself may be affected by network nodes intentionally altering the bits of the AltMark Option or injecting Options headers as a means for Denial of Service (DoS), the Alternate Marking MUST be applied in the context of a controlled domain, where the network nodes are locally administered and this type of attack can be avoided. For this reason, the implementation of the method is not done on the end node if it is not fully managed and does not belong to the controlled domain. Packets generated outside the controlled domain may consume router resources by maliciously using the HbH Option, but this can be mitigated by filtering these packets at the controlled domain boundary. This can be done because, if the end node does not belong to the controlled domain, it is not supposed to add the AltMark HbH Option, and it can be easily recognized.

An attacker that does not belong to the controlled domain can maliciously send packets with AltMark Option. But if Alternate Marking is not supported in the controlled domain, no problem happens because the AltMark Option is treated as any other unrecognized option and will not be considered by the nodes since they are not configured to deal with it, so the only effect is the increased MTU (by 48 bits). While if Alternate Marking is supported in the controlled domain, it is also necessary to avoid that the measurements are affected and external packets with AltMark Option MUST be filtered. As any other Hop-by-Hop Options or Destination Options, it is possible to filter AltMark Options entering or leaving the domain e.g. by using ACL extensions for filtering.

The flow identifier (FlowMonID) composes the AltMark Option together with the two marking bits (L and D). As explained in Section 5.3, there is a chance of collision if the FlowMonID is set pseudo randomly and a solution exists. In general this may not be a problem and a low rate of ambiguous FlowMonIDs can be acceptable, since this does not cause significant harm to the operators or their clients and this harm may not justify the complications of avoiding it. But, for large scale measurements, a big number of flows could be monitored and the probability of a collision is higher, thus the disambiguation of the FlowMonID field can be considered.

The privacy concerns also need to be analyzed even if the method only relies on information contained in the Option Header without any release of user data. Indeed, from a confidentiality perspective, although AltMark Option does not contain user data, the metadata can be used for network reconnaissance to compromise the privacy of users by allowing attackers to collect information about network performance and network paths. AltMark Option contains two kinds of metadata: the marking bits (L and D bits) and the flow identifier (FlowMonID).
The marking bits are the small information that is exchanged between the network nodes. Therefore, due to this intrinsic characteristic, network reconnaissance through passive eavesdropping on data-plane traffic is difficult. Indeed, an attacker cannot gain information about network performance from a single monitoring point. The only way for an attacker can be to eavesdrop on multiple monitoring points at the same time, because they have to do the same kind of calculation and aggregation as Alternate Marking requires.

The FlowMonID field is used in the AltMark Option as the identifier of the monitored flow. It represents a more sensitive information for network reconnaissance and may allow a flow tracking type of attack because an attacker could collect information about network paths.

Furthermore, in a pervasive surveillance attack, the information that can be derived over time is more. But, as further described hereinafter, the application of the Alternate Marking to a controlled domain helps to mitigate all the above aspects of privacy concerns.

At the management plane, attacks can be set up by misconfiguring or by maliciously configuring AltMark Option. Thus, AltMark Option configuration MUST be secured in a way that authenticates authorized users and verifies the integrity of configuration procedures. Solutions to ensure the integrity of AltMark Option are outside the scope of this document. Also, attacks on the reporting of the statistics between the monitoring points and the network management system (e.g. centralized controller) can interfere with the proper functioning of the system. Hence, the channels used to report back flow statistics MUST be secured.

As stated above, the precondition for the application of the Alternate Marking is that it MUST be applied in specific controlled domains, thus confining the potential attack vectors within the network domain. [RFC8799] analyzes and discusses the trend towards network behaviors that can be applied only within a limited domain. This is due to the specific set of requirements especially related to security, network management, policies and options supported which may vary between such limited domains. A limited administrative domain provides the network administrator with the means to select, monitor and control the access to the network, making it a trusted domain. In this regard it is expected to enforce policies at the domain boundaries to filter both external packets with AltMark Option entering the domain and internal packets with AltMark Option leaving the domain. Therefore, the trusted domain is unlikely subject to hijacking of packets since packets with AltMark Option are processed and used only within the controlled domain.
As stated, the application to a controlled domain ensures the control over the packets entering and leaving the domain, but despite that, leakages may happen for different reasons, such as a failure or a fault. In this case, nodes outside the domain MUST simply ignore packets with AltMark Option since they are not configured to handle it and should not process it.

Additionally, it is to be noted that the AltMark Option is carried by the Options Header and it may have some impact on the packet sizes for the monitored flow and on the path MTU, since some packets might exceed the MTU. However, the relative small size (48 bit in total) of these Option Headers and its application to a controlled domain help to mitigate the problem.

It is worth mentioning that the security concerns may change based on the specific deployment scenario and related threat analysis, which can lead to specific security solutions that are beyond the scope of this document. As an example, the AltMark Option can be used as Hop-by-Hop or Destination Option and, in case of Destination Option, multiple administrative domains may be traversed by the AltMark Option that is not confined to a single administrative domain. In this case, the user, aware of the kind of risks, may still want to use Alternate Marking for telemetry and test purposes but the controlled domain must be composed by more than one administrative domains. To this end, the inter-domain links need to be secured (e.g., by IPsec, VPNs) in order to avoid external threats and realize the whole controlled domain.

It might be theoretically possible to modulate the marking or the other fields of the AltMark Option to serve as a covert channel to be used by an on-path observer. This may affect both the data and management plane, but, here too, the application to a controlled domain helps to reduce the effects.

The Alternate Marking application described in this document relies on a 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]. Network Time Security (NTS), described in [RFC8915], is a mechanism that can be employed. Also, the time, which is distributed to the network nodes through the time protocol, is centrally taken from an external accurate time source, such as an atomic clock or a GPS clock. By attacking the time source it can be possible to compromise the integrity of the measurement as well. There are security measures that can be taken to mitigate the GPS spoofing attacks and a network administrator should certainly employ solutions to secure the network domain.
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 assignment 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/).

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Binary Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>00 0 tbd</td>
<td>AltMark</td>
<td>[This draft]</td>
</tr>
</tbody>
</table>

8. Acknowledgements

The authors would like to thank Bob Hinden, Ole Troan, Martin Duke, Lars Eggert, Roman Danyliw, Alvaro Retana, Eric Vyncke, Warren Kumari, Benjamin Kaduk, Stewart Bryant, Christopher Wood, Yoshifumi Nishida, Tom Herbert, Stefano Previdi, Brian Carpenter, Greg Mirsky, Ron Bonica for the precious comments and suggestions.

9. References

9.1. Normative References

[I-D.ietf-ippm-rfc8321bis]

[I-D.ietf-ippm-rfc8889bis]


9.2. Informative References

[I-D.chen-pce-pcep-ifit]

[I-D.fz-spring-srv6-alt-mark]

[I-D.hinden-6man-hbh-processing]

[I-D.ietf-idr-sr-policy-ifit]

[I-D.peng-v6ops-hbh]


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
IPv6 Minimum Path MTU Hop-by-Hop Option
draft-ietf-6man-mtu-option-15

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 11 November 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

This document specifies a new IPv6 Hop-by-Hop (HBH) Option to record the minimum Maximum Transmission Unit (MTU) along the forward path between a source and a destination host. The source host creates a packet with this option and initializes 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 input to the method that sets the Path MTU (PMTU) used by upper layer protocols.

The IPv6 Minimum Path MTU Hop-by-Hop (MinPMTU HBH) Option is designed to work with packet sizes that can be specified in the IPv6 header. The maximum packet size that can be specified in an IPv6 header is 65,535 octets ($2^{16}$).

This method has the potential to complete Path MTU discovery in a single round trip time, even over paths that have successive links each with a lower MTU.

The mechanism defined in this document is focused on Unicast, it does not describe Multicast. That is left for future work.

1.1. Example Operation

The figure below illustrates the operation of the method. In this case, the path between the source host and the destination host comprises three links, the source 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.

![Diagram of network path with MTUs](image)

Three scenarios are described:

* Scenario 1, considers all links to have an 9000 byte MTU and the method is supported by both routers. The initial Min-PMTU is not modified along the path, and therefore the PMTU is 9000 bytes.

* Scenario 2, considers the link between R2 and 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 Too Big (PTB) message had not been delivered to the sender [RFC8201].

These scenarios are summarized in the table below. "H" in R1 and/or R2 columns means the router understands the MinPMTU HBH option.

<table>
<thead>
<tr>
<th></th>
<th>MTU-S</th>
<th>MTU-D</th>
<th>R1</th>
<th>R2</th>
<th>Rec PMTU</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9000B</td>
<td>9000B</td>
<td>H</td>
<td>H</td>
<td>9000 B</td>
<td>Endpoints attempt to use a 9000 B PMTU.</td>
</tr>
<tr>
<td>2</td>
<td>9000B</td>
<td>1500B</td>
<td>H</td>
<td>H</td>
<td>1500 B</td>
<td>Endpoints attempt to use a 1500 B PMTU.</td>
</tr>
<tr>
<td>3</td>
<td>9000B</td>
<td>1500B</td>
<td>H</td>
<td>-</td>
<td>9000 B</td>
<td>Endpoints attempt to use a 9000 B PMTU,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>but need to implement a method to fall back to discover and use a 1500 B PMTU.</td>
</tr>
</tbody>
</table>

Figure 2

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

IPv6 as specified in [RFC8200] allows nodes to optionally process the Hop-by-Hop header. 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. It fails to work in various cases, including when nodes in the middle of the network do not send ICMPv6 PTB messages, or rate-limited ICMPv6 messages, or do not have a return path to the source host.

This results in many transport layer 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 send large packets are forced to use IPv6 Fragmentation [RFC8200], which can reduce the reliability of Internet communication [RFC8900].

Encapsulations and network-layer tunnels further reduce the payload size available for a transport protocol 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 larger packets can improve host performance, e.g., avoiding limits to packet processing by the packet rate. For example, the packet per second rate required to reach wire speed on a 10G link with 1280 byte packets is about 977K packets per second (pps), vs. 139K pps for 9000 byte packets.
The purpose of this document 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.

A similar mechanism was proposed in 1988 for IPv4 in [RFC1063] by Jeff Mogul, C. Kent, Craig Partridge, and Keith McCloughire. It was later obsoleted in 1990 by [RFC1191], the current deployed approach to Path MTU Discovery. In contrast, the method described in this document uses the Hop-by-Hop option of IPv6. It does not replace PMTUD [RFC8201], PLPPMTUD [RFC4821] or Datagram PLPMTUD [RFC8899], but rather is designed to compliment these methods.

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

The Path MTU option is designed for environments where there is control over the hosts and nodes that connect them, and where there is more than one MTU size in use. For example, in Data Centers and on paths between Data Centers, to allow hosts 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 can be executed on a router’s fast path. A successful experiment depends on both implementation by host and router vendors and deployment by operators. The contained use-case of connections within and between Data Centers could be a driver for deployment.

The method could also be useful in other environments, including the general Internet, and offers advantage when this Hop-by-Hop Option is supported on all paths. The method is more robust when used to probe the path using packets that do not carry application data and when also paired with a method such as Packetization Layer PMTUD [RFC4821] or Datagram PLPMTUD [RFC8899].

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

The Minimum Path MTU Hop-by-Hop Option has the following format:
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 value field in Option Data field supports PMTU values from 0 to 65,534 octets, the maximum size represented by the Path MTU option.

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] MUST 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 when the destination host is to send a PMTU Option back to the source host.

Figure 3

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. This encoding fits in the minimum-sized Hop-by-Hop Option header.
6. Router, Host, and Transport Layer Behaviors

6.1. Router Behavior

Routers that are not configured to support Hop-by-Hop Options are not expected to examine or process the contents of this option [RFC8200].

Routers that support Hop-by-Hop Options, but are not configured to support this option SHOULD skip over this option and continue to processing the header [RFC8200].

Routers that support this option MUST 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. (The router processing is performed without checking the valid range of the Min-PMTU or the Rtn-PMTU fields.)

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

6.2. Host Operating System Behavior

The PMTU entry associated with the destination in the host’s destination cache [RFC4861] SHOULD be updated after detecting a change using the IPv6 Minimum Path MTU Hop-by-Hop Option. This cached value can be used by other flows that share the host’s destination cache.

The value in the host destination cache SHOULD be used by PLPMTUD to select an initial PMTU for a flow. The cached PMTU is only increased by PLPMTUD when the Packetization Layer determines the path actually supports a larger PMTU [RFC4821] [RFC8899].

When requested to send an IPv6 packet with the MinPMTU HBH option, the source host includes the option in an outgoing packet. The source host MUST 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.

When a host includes the option in a packet it sends, the host SHOULD set the Rtn-PMTU field to the previously cached value of the received Minimum Path MTU for the flow in the Rtn-PMTU field (see Section 6.3.3). If this value is not set (for example, because there is no cached reported Min-PMTU value), the Rtn-PMTU field value 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. The R-Flag SHOULD NOT be set when the MinPMTU HBH Option was sent solely to provide requested feedback on the return Path MTU to avoid each response generating another response.

The destination host controls when to send a packet with this option in response to an R-flag, as well as which packets to include it in. The destination host MAY limit the rate at which it sends these packets.

A destination host only sets the R Flag if it wishes the source host to also return the discovered PMTU value for the path from the destination to the source.

The normal sequence of operation of the R-Flag using the terminology from the diagram in Figure 1 is:

1. The source sends a probe to the destination. The sender sets the R-Flag.
2. The destination responds by sending a probe including the received Min-PMTU as the Rtn-PMTU. A destination that does not wish to probe the return path sets the R-Flag to 0.

6.3. Transport Layer Behavior

This Hop-by-Hop option is intended to be used with a path MTU discovery method.

PLPMTUD [RFC9000] uses probe packets for two distinct functions:

* Probe packets are used to confirm connectivity. Such probes can be of any size up to the PLPMTU. These probe packets are sent to solicit a response use the path to the remote node. These probe packets can carry the Hop-by-Hop PMTU option, providing the final size of the packet does not exceed the current PLPMTU. After validating that the packet originates from the path (section 4.6.1), the PLPMTUD method can use the reported size from the Hop-by-Hop option as the next search point when it resumes the search algorithm. (This use resembles the use of the PTB_SIZE information in section 4.6.2 of [RFC8899]

* A second use of probe packets is to explore if a path supports a packet size greater than the current PLPMTU. If this probe packet is successfully delivered (as determined by the source host), then the PLPMTU is raised to the size of the successful probe. These probe packets do not usually set the Path MTU Hop-by-Hop option.
See section 1.2 of [RFC8899]. Section 4.1 of [RFC8899] also describes ways that a Probe Packet can be constructed, depending on whether the probe packets carry application data.

* The PMTU Hop-by-Hop Option Probe can be sent on packets that include application data, but needs to be robust to potential loss of the packet (i.e., with the possibility that retransmission might be needed if the packet is lost).

* Using a PMTU Probe on packets that do not carry application data will avoid the need for loss recovery if a router on the path drops packets that set this option. (This avoids the transport needing to retransmit a lost packet that includes this option.) This is the normal default format for both uses of probes.

6.3.1. Including the Option in an Outgoing Packet

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

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 a packet that carries a TCP segment sent after the connection is established. A segment without data could be used, to avoid the need to retransmit this data if the probe packet is lost. The discovered value can be used to inform PLPMTUD [RFC4821].

NOTE: A TCP SYN 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. If this option were to be included in a TCP SYN, it could increase the probability that the SYN segment is lost when routers on the path drop packets with this option (see Section 6.3.6), which could have an unwanted impact on the result of racing options [I-D.ietf-taps-arch] or feature negotiation.
* 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 MinPMTU HBH option can be used with ICMPv6 [RFC4443]. This requires a response from the remote node and therefore is restricted to use with ICMPv6 echo messages. The MinPMTU HBH option could provide additional information about the PMTU that might be supported by a path. This could be use as a diagnostic tool to measure the PMTU of a path. As with other uses, the actual supported PMTU is only confirmed after receiving a response to a subsequent probe of the PMTU size.

* A datagram transport can utilise DPLPMTUD [RFC8899]. For example, QUIC (see section 14.3 of [RFC9000]), can use DPLPMTUD to determine whether the path to a destination will support a desired maximum datagram size. When using the IPv6 MinPMTU HBH option, the option could be added to an additional QUIC PMTU Probe that is of minimal size (or one no larger than the currently supported PMTU size). Once the return Path MTU value in the MinPMTU HBH option has been learned, DPLPMTUD can be triggered to test for a larger PLPMTU using an appropriately sized PLPMTU Probe Packet (see section 5.3.1 of [RFC8899]).

* The use of this option with DNS and DNSSEC over UDP is expected 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.

* This method also can be used with Anycast to discover the PMTU of the path, but the use needs to be aware that the Anycast binding might change.
6.3.2. Validation of the Packet that includes the Option

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. This validates that the packet forms a part of an existing flow, using context available at the upper layer. 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, see Section 5.1 of [RFC8085]. 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 MinPMTU HBH option without further processing.

6.3.3. Receiving the Option

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 MinPMTU HBH Option with the R-Flag SHOULD include the MinPMTU HBH 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 loses state.

Discussion:

* Some upper layer protocols send packets less frequently than the rate at which 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 source and destination hosts. 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 the 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 there is a layer-2 device with a lower MTU.

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

* The path might have changed between the time when the probe was sent and when the Rtn-PMTU value received.

IPv6 requires that every link in the Internet have an MTU of 1280 octets or greater. A node MUST ignore a Rtn-PMTU value less than 1280 octets [RFC8200].

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 DPLFMTUD [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 to be maintained for each transport endpoint, or for each network address.
6.3.5. Detecting Path Changes

Path characteristics can change and the actual PMTU could increase or decrease over time. For instance, following a path change when packets are forwarded over a link with a different MTU than that previously used. To bound the delay in discovering an increase in the actual PMTU, a host with a link MTU larger than the current PMTU SHOULD periodically send the MinPMTU HBH 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.

6.3.6. 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 can 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

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

IANA is requested to update this registry to point to this document and remove the Temporary status.

8. Security Considerations

This section discusses the security considerations. It first reviews router option processing. It then 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. Router Option Processing

This option shares the characteristics of all other IPv6 Hop-by-Hop Options, in that if not supported at line rate it could be used to degrade the performance of a router. This option, while simple, is no different to other uses of IPv6 Hop-by-Hop options.

It is common for routers to ignore the Hop-by-Hop Option header or drop packets containing a Hop-by-Hop Option header. Routers implementing IPv6 according to [RFC8200] only examine and process the Hop-by-Hop Options header if explicitly configured to do so.

8.2. Network Layer Host Processing

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

For comparison, the ICMPv6 Packet Too Big message used in [RFC8201] Path MTU Discovery, the source host has an inherent trust relationship with the destination host including this option. This trust relationship can be used to help verify the option. ICMPv6 Packet Too Big messages are sent from any router on the path to the destination host, the source host has no prior knowledge of these routers (except for the first hop router).

Reception of this packet will require 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 applies 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.
8.3. 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 MinPMTU HBH 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. Note that AH style authentication [RFC4302] while authenticating the payload and outer IPv6 header, does not check Hop-by-Hop options that change on route.

8.4. 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 host 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 causes the destination to add the option to a packet sent to the source host. 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 MinPMTU HBH option is received. This attack could be mitigated by limiting the sending of the MinPMTU HBH option in reply to incoming packets that carry the option.
8.5. 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 (see Section 8.3). 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 MinPMTU HBH option corresponding to the flow.

* Processing of the option could cause a destination host to add the MinPMTU HBH 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.

8.6. 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 6.3.6.

When a forged packet causes a packet to be sent including the MinPMTU HBH option, and the return path does not forward packets with this option, the packet will be dropped Section 6.3.6. 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 an R-Flag by including the option in a packet that does not carry application data.

9. Experiment Goals

This section describes the experimental goals of this specification.
A successful deployment of the method depends upon several components being implemented and deployed:

* Support in the sending node (see Section 6.2). This also requires corresponding support in upper layer protocols (see Section 6.3).

* Router support in nodes (see Section 6.1). The IETF continues to provide recommendations on the use of IPv6 Hop-by-Hop options, for example Section 2.2.2 of [RFC9099]. This document does not update the way router implementations configure support for Hop-by-Hop options.

* Support in the receiving node (see Section 6.3.3).

Experience from deployment is an expected input to any decision to progress this specification from Experimental to IETF Standards Track. Appropriate inputs might include:

* Reports of implementation experience;

* Measurements of the number paths where the method can be used;

* Measurements showing the benefit realized or the implications of using specific methods over specific paths.

10. Implementation Status

At the time this document was published there are two known implementations of the Path MTU Hop-by-Hop option. These are:

* Wireshark dissector. This is shipping in production in Wireshark version 3.2 [WIRESHARK].

* A prototype in the open source version of the FD.io Vector Packet Processing (VPP) technology [VPP]. At the time this document was published, the source code can be found [VPP_SRC].

11. Acknowledgments

Helpful comments were received from Tom Herbert, Tom Jones, Fred Templin, Ole Troan, Tianran Zhou, Jen Linkova, Brian Carpenter, Peng Shuping, Mark Smith, Fernando Gont, Michael Dougherty, Erik Kline, and other members of the 6MAN working group.

12. Change log [RFC Editor: Please remove]

draft-ietf-6man-mtu-option-15, 2022-May-10
* Correcting an editing mistake in Appendix A.
* Editorial Change.

draft-ietf-6man-mtu-option-14, 2022-April-15

* Area Director Reviews:
  - Lars Eggert’s Review: Fixed "nits".
  - Eric Vyncke’s Review: Added that this work is focused on Unicast, removed Discussion from Section 6.1, revised text on PLPMTUD probing, changed SHOULD to MUST in Section 6.3.4, and fixed several NITs.
  - Alvaro Retana’s Review: Changed SHOULD language to more general text in Section 6.1
  - ARTART Review: Added new Appendix "Examples of Usage" with diagrams showing examples of use.
* Editorial Changes.

draft-ietf-6man-mtu-option-13, 2022-February-28

* Area Directorate Reviews:
  - SECDIR Review: Fixed "nit".
  - TSVART Review: Restructured Section 6 including making Transport Behavior more prominent, added text about ICMPv6 to Section 6.3.1, moved the text about prior work in RFC1063 to Section 2.
  - GENART Review: Added text to Section 1 that this option was designed to work with packet sizes that can be specified in the IPv6 Header.
* Editorial Changes.

draft-ietf-6man-mtu-option-12, 2022-January-26

* Clarified a few issues raised by AD review by Erik Kline AD review.

draft-ietf-6man-mtu-option-11, 2021-September-30

* Clarifications and editorial changes to the Security Considerations section based on early AD review by Erik Kline.

draft-ietf-6man-mtu-option-10, 2021-September-27

* Clarifications and editorial changes based on second chair review by Ole Troan.
* Editorial changes.
* Clarifications and editorial changes based on review by Michael Dougherty.

* Clarifications and editorial changes based on chair review by Ole Troan.
* Correction and clarifications based on review by Fernando Gont.

* Added Experiment Goals section.
* Added Implementation Status section.
* Updated the IANA Considerations section to point to this document and remove Temporary status.
* Clarifications and editorial changes based on review by Mark Smith.

* Transport usage of the mechanism clarified in response to feedback and suggestions from Jen Linkova.
* Restructured Section 6 to improve readability.
* Editorial changes.

* Editorial changes.

* Fixes for typos.

* 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.

* 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.
* ICMPv6 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.

13. References

13.1. Normative References

13.2. Informative References


Appendix A. Examples of Usage

This section provides examples that illustrate a use of the MinPMTU HBH option by a source using DPLPMTUD to discover the PLPMTU supported by a path. They consider a path where the on-path router has been configured with an outgoing MTU of d’. The source starts by transmission of packets of size a, and then uses DPLPMTUD to seek to increase the size in steps resulting in sizes of b, c, d, e, etc., (chosen by the search algorithm used by DPLPMTUD). The search algorithm terminates with a PLPMTU that is at least d and is less than or equal to d’.

The first example considers DPLPMTUD without using the MinPMTU HBH option. In this case, DPLPMTUD searches using an increasing size of probe packet. Probe packets of size (e) are sent, which are larger than the actual PMTU. In this example, PTB messages are not received from the routers and repeated unsuccessful probes result in the search phase completing. Packets of data are never sent with a size larger than the size of the last confirmed probe packet. ACKs of data packets are not shown.
The second example considers DPLPMTUD with the MinPMTU HBH option set on a connectivity probe packet.

The IPv6 option is sent end-to-end, and the Min-PMTU is updated by a router on the path to d', which is returned in a response that also sets the MinPMTU HBH option. Upon receiving Rtn-PMTU value is received, DPLPMTUD immediately sends a probe packet of the target size (d'). If the probe packet is confirmed for the path, the PLPMTU is updated, allowing the source to use data packets up to size d'. (The search algorithm is allowed to continue to probe to see if the path supports a larger size.) Packets of data are never sent with a size larger than the last confirmed probe size, d'.

```
----Packets of data size (a) ---------------------------------->
----Probe size (b) ------------------------------------------>
<---------------------------------- ACK of probe ---------->
----Packets of data size (b) ---------------------------------->
----Probe size (c) ------------------------------------------>
<---------------------------------- ACK of probe ---------->
----Packets of data size (c) ---------------------------------->
----Probe size (d) ------------------------------------------>
<---------------------------------- ACK of probe ---------->
----Packets of data size (d) ---------------------------------->
<---------------------------------- ACK of probe ---------->
...
----Probe size (e) ---------X
  X----ICMPv6 PTB (d') --|
----Packets of data size (d) ---------------------------------->
----Probe size (e) ---------X (again)
  X----ICMPv6 PTB (d') --|
----Packets of data size (d) ---------------------------------->
...
```

etc, until MaxProbes are unsuccessful and search phase completes.

```
----Packets of data size (d) ---------------------------------->
```

Figure 4

etc, until MaxProbes are unsuccessful and search phase completes.

```
----Packets of data size (d) ---------------------------------->
```

The second example considers DPLPMTUD with the MinPMTU HBH option set on a connectivity probe packet.

The IPv6 option is sent end-to-end, and the Min-PMTU is updated by a router on the path to d’, which is returned in a response that also sets the MinPMTU HBH option. Upon receiving Rtn-PMTU value is received, DPLPMTUD immediately sends a probe packet of the target size (d’). If the probe packet is confirmed for the path, the PLPMTU is updated, allowing the source to use data packets up to size d’. (The search algorithm is allowed to continue to probe to see if the path supports a larger size.) Packets of data are never sent with a size larger than the last confirmed probe size, d’.

```
----Packets of data size (a) ---------------------------------->
----Connectivity probe with MinPMTU-
  +--updated to minPMTU=d’----->
<-----------------ACK with Rtn-PMTU=d’------------------->
----Packets of data size (a) ---------------------------------->
----Probe size (d’) ------------------------------------------>
<---------------------------------- ACK of probe ---------->
----Packets of data size (d’) ---------------------------------->
Search phase completes.
----Packets of data size (d’) ---------------------------------->
```

```
----Packets of data size (d’) ---------------------------------->
```

Hinden & Fairhurst Expires 11 November 2022 [Page 25]
The final example considers DPLPMTUD with the MinPMTU HBH option set on a connectivity probe packet, but shows the effect when this connectivity probe packet is dropped.

In this case, the packet with the MinPMTU HBH option is not received. DPLPMTUD searches using probe packets of increasing size, increasing the PLPMTU when the probes are confirmed. An ICMPv6 PTB message is received when the probed size exceeds the actual PMTU, indicating a PTB_SIZE of \( d' \). DPLPMTUD immediately sends a probe packet of the target size (\( d' \)). If the probe packet is confirmed for the path, the PLPMTU is updated, allowing the source to use data packets up to size \( d' \). If the ICMPv6 PTB message is not received, the DPLPMTU will be the last confirmed probe size, \( d \).

---Packets of data size (a) --
---Connectivity probe with MinPMTU --
---Packets of data size (a) --
---Probe size (b) --
<---------------------------------- ACK of probe --
---Packets of data size (b) --
---Probe size (c) --
<---------------------------------- ACK of probe --
---Packets of data size (c) --
---Probe size (d) --
<---------------------------------- ACK of probe --
---Packets of data size (d) --
---Probe size (e) --
<--ICMPv6 PTB PTB_SIZE(d') --
---Packets of data size (d) --
---Probe size (d') using target set by PTB_SIZE --
<---------------------------------- ACK of probe --
Search phase completes.
---Packets of data size (d') --

The number of probe rounds depends on the number of steps needed by the search algorithm, and is typically larger for a larger PMTU.

Authors’ Addresses

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, CA 94070
United States of America
Improving the Robustness of Stateless Address Autoconfiguration (SLAAC) to Flash Renumbering Events
draft-ietf-6man-slaac-renum-03

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

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

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

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

This Internet-Draft will expire on 8 December 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components
Table of Contents

1. Introduction .................................................. 3
2. Terminology .................................................... 3
3. SLAAC reaction to Flash-renumbering Events .................. 4
   3.1. Renumbering without Explicit Signaling ................. 4
   3.2. Renumbering with Explicit Signaling ................... 5
4. Improvements to Stateless Address Autoconfiguration (SLAAC) ........................................ 6
   4.1. More Appropriate Lifetime Values ....................... 7
       4.1.1. Router Configuration Variables .................... 7
   4.2. Honor Small PIO Valid Lifetimes ....................... 8
   4.3. Interface Initialization .................................. 9
   4.4. Conveying Information in Router Advertisement (RA) Messages ........................................ 10
   4.5. Recovery from Stale Configuration Information without Explicit Signaling ............... 10
5. IANA Considerations ............................................ 10
6. Implementation Status ......................................... 10
   6.1. More Appropriate Lifetime Values ....................... 10
       6.1.1. Router Configuration Variables .................... 10
   6.2. Honor Small PIO Valid Lifetimes ....................... 11
       6.2.1. Linux Kernel ........................................ 11
       6.2.2. NetworkManager ..................................... 11
   6.3. Conveying Information in Router Advertisement (RA) Messages ........................................ 11
   6.4. Recovery from Stale Configuration Information without Explicit Signaling ............... 11
       6.4.1. dhcpcd(8) ............................................ 11
   6.5. Other mitigations implemented in products ................ 11
7. Security Considerations ....................................... 12
8. Acknowledgments ............................................... 12
9. References ..................................................... 13
   9.1. Normative References .................................... 13
   9.2. Informative References .................................. 14
Appendix A. Analysis of Some Suggested Workarounds ............ 15
   A.1. On a Possible Reaction to ICMPv6 Error Messages ..... 16
   A.2. On a Possible Improvement to Source Address Selection .. 16
Authors’ Addresses ............................................... 18
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 [RFC8978]. 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. [RFC8978] 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:

* Avoiding the use of stale addresses for new communication instances

* 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].
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:

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

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

* 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 is 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:

\[
\text{AdvPreferredLifetime: } \max(\text{AdvDefaultLifetime}, 3 \times \text{MaxRtrAdvInterval})
\]

\[
\text{AdvValidLifetime: } 2 \times \text{AdvPreferredLifetime}
\]

where:

\text{AdvPreferredLifetime:} \\
Value to be placed in the "Preferred Lifetime" field of the PIO.

\text{AdvValidLifetime:} \\
Value to be placed in the "Valid Lifetime" field of the PIO.

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

\text{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:

[RFC9096] 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 [RFC8978] 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]:

* Preferred Lifetime: 14400 seconds (4 hours)

* Valid Lifetime: 86400 seconds (1 day)
This is not following the specific recommendation in this document, but 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:

* Rather than recording prefixes on stable storage (as recommended in [RFC9096]), 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 of 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 [RFC8978], 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


9.2. Informative References


Appendix A. Analysis of Some Suggested Workarounds

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:

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

* 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.

* 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.

* 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:
* 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.

* 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.

* [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:

* 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.

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

* 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
Self-configuring Stub Networks: Problem Statement
draft-lemon-stub-networks-ps-02

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

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

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

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

This Internet-Draft will expire on 27 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

* 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.

* 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.

* 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:

* is discoverable by applicable hosts that are not on the stub network

* is able to acquire an IP address that can be used to communicate with applicable hosts not on the stub network

* has reachability to the network(s) to which applicable hosts are attached

* 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 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

* Infrastructure network must support prefix allocation using DHCPv6 prefix delegation.

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

* 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:

* The stub host can use Port Control Protocol to register a port, and then advertise that using SRP.

* 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


Author’s Address

Ted Lemon
Apple, Inc.
One Apple Park Way
Cupertino, California 95014
United States of America
Email: mellon@fugue.com
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

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

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

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

7.2. Informative References


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
Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces
draft-templin-6man-omni-interface-99

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 presented that allows mobile nodes to coordinate with a network-based mobility service and/or with other mobile node peers. This document specifies the transmission of IP packets over Overlay Multilink Network (OMNI) Interfaces.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 25, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents.
Table of Contents

1. Introduction ................................................. 3
2. Terminology ................................................... 5
3. Requirements .................................................. 10
4. Overlay Multilink Network (OMNI) Interface Model ................. 11
5. OMNI Interface Maximum Transmission Unit (MTU) .................. 17
6. The OMNI Adaptation Layer (OAL) .............................. 18
   6.1. OAL Source Encapsulation and Fragmentation ................. 18
   6.2. OAL *NET Encapsulation and Re-Encapsulation ............... 23
   6.3. OAL Destination Decapsulation and Reassembly ............... 24
   6.4. OAL Header Compression .................................. 25
   6.5. OAL Fragment Identification Window Maintenance ............. 28
   6.6. OAL Fragment Retransmission ................................ 29
   6.7. OAL MTU Feedback Messaging ................................ 30
   6.8. OAL Requirements ........................................ 32
   6.9. OAL Fragmentation Security Implications ..................... 33
   6.10. OAL Super-Packets ...................................... 34
7. Frame Format ................................................. 36
8. Link-Local Addresses (LLAs) .................................. 36
9. Unique-Local Addresses (ULAs) ................................ 38
10. Global Unicast Addresses (GUAs) .............................. 39
11. Node Identification .......................................... 40
12. Address Mapping - Unicast .................................... 40
    12.1. Sub-Options ........................................... 42
      12.1.1. Pad1 .............................................. 44
      12.1.2. PadN .............................................. 45
      12.1.3. Interface Attributes (Type 1) ......................... 45
      12.1.4. Interface Attributes (Type 2) ......................... 47
      12.1.5. Traffic Selector .................................... 51
      12.1.6. MS-Register ........................................ 51
      12.1.7. MS-Release ......................................... 52
      12.1.8. Geo Coordinates ..................................... 53
      12.1.10. Host Identity Protocol (HIP) Message ................. 54
      12.1.11. Reassembly Limit .................................... 55
      12.1.12. Fragmentation Report ................................ 57
      12.1.13. Node Identification ................................ 58
      12.1.14. Sub-Type Extension ................................ 60
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-line 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 Interface (OMNI)") 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 internally employs the "OMNI Adaptation Layer (OAL)" to ensure that original IP packets are delivered without loss due to size restrictions. 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 15).

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 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 within the same OMNI domain 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 interface within the same OMNI domain configures a common ULA prefix [ULA]::/48, and configures a unique 16-bit Subnet ID '*' to construct the sub-prefix [ULA*]::/64 (see: Section 9). 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 have been proposed. 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 OAL operates as a sublayer between L3 and L2 based on IPv6 encapsulation [RFC2473] as discussed in the following sections. OMNI interfaces enable Multilink, Mobility, Multihop, Multicast and MTU services (i.e., the "five M’s"), with provisions for both Vehicle-to-Infrastructure (V2I) communications and Vehicle-to-Vehicle (V2V) communications outside the context of infrastructure.

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 (MNP) 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 10). 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. Note that OMNI link Relay nodes may also service non-MNP routes (i.e., GUA prefixes not covered by an MSP) but that these correspond to fixed correspondent nodes and not MNs. Other than this distinction, MNP
and non-MNP routes are treated exactly the same by the OMNI routing system.

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 sublayer service whereby original IP packets admitted into the interface are wrapped in an IPv6 header and subject to fragmentation and reassembly. 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.

original IP packet
a whole IP packet or fragment admitted into the OMNI interface by the network layer prior to OAL encapsulation and fragmentation, or an IP packet delivered to the network layer by the OMNI interface following OAL decapsulation and reassembly.

OAL packet
an original IP packet encapsulated in OAL headers and trailers before OAL fragmentation, or following OAL reassembly.

OAL fragment
a portion of an OAL packet following fragmentation but prior to *NET encapsulation, or following *NET encapsulation but prior to OAL reassembly.

(OAL) atomic fragment
an OAL packet that does not require fragmentation is always encapsulated as an "atomic fragment" with a Fragment Header with Fragment Offset and More Fragments both set to 0, but with a valid Identification value.

(OAL) carrier packet
an encapsulated OAL fragment following *NET encapsulation or prior to *NET decapsulation. OAL sources and destinations exchange carrier packets over underlying interfaces, and may be separated by one or more OAL intermediate nodes. OAL intermediate nodes may perform re-encapsulation on carrier packets by removing the *NET headers of the first hop network and replacing them with new *NET headers for the next hop network.

OAL source
an OMNI interface acts as an OAL source when it encapsulates original IP packets to form OAL packets, then performs OAL fragmentation and *NET encapsulation to create carrier packets.

OAL destination
an OMNI interface acts as an OAL destination when it decapsulates carrier packets, then performs OAL reassembly and decapsulation to derive the original IP packet.

OAL intermediate node
an OMNI interface acts as an OAL intermediate node when it removes the *NET headers of carrier packets received on a first segment, then re-encapsulates the carrier packets in new *NET headers and forwards them into the next segment.

OMNI Option
an IPv6 Neighbor Discovery option providing multilink parameters for the OMNI interface as specified in Section 12.

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 8.

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 8.

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.

Multihop
an iterative relaying of IP packets between MNs over an OMNI underlying interface technology (such as omnidirectional wireless) without support of fixed infrastructure. Multihop services entail node-to-node relaying within a Mobile/Vehicular Ad-hoc Network (MANET/VANET) for MN-to-MN communications and/or for "range extension" where MNs within range of communications infrastructure elements provide forwarding services for other MNs.
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 8). 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 9.

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.

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 virtual interface configured over one or more underlying interfaces, which may be physical (e.g., an aeronautical radio link, etc.) or virtual (e.g., an Internet or higher-layer "tunnel"). 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 nexus for multiple underlying interfaces that appear as L2 communication channels in the architecture.

![Figure 1: OMNI Interface Architectural Layering Model](image)

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

- 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.

- 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.

- 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.

- 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 interface forwards original IP packets from the network layer (L3) using the OMNI Adaptation Layer (OAL) (see: Section 5) as an encapsulation and fragmentation sublayer service. This "OAL source" then further encapsulates the resulting OAL packets/fragments in *NET headers to create OAL carrier packets for transmission over underlying interfaces (L2/L1). The target OMNI interface receives the carrier packets from underlying interfaces (L1/L2) and discards the *NET headers. If the resulting OAL packets/fragments are addressed to itself, the OMNI interface acts as an "OAL destination" and performs reassembly if necessary, discards the OAL encapsulation, and delivers the original IP packet to the network layer (L3). If the OAL fragments are addressed to another node, the OMNI interface instead acts as an "OAL intermediate node" by re-encapsulating in new *NET headers and forwarding the new carrier packets over an underlying interface without reassembling or discarding the OAL encapsulation. The OAL source and OAL destination are seen as "neighbors" on the OMNI link, while OAL intermediate nodes are seen as "bridges".

The OMNI interface can send/receive original IP packets to/from underlying interfaces while including/omitting various encapsulations including OAL, UDP, IP and L2. The network layer can also access the underlying interfaces directly while bypassing the OMNI interface entirely when necessary. This architectural flexibility may be beneficial for underlying interfaces (e.g., some aviation data links) for which encapsulation overhead may be a primary consideration. OMNI interfaces that send original IP packets directly over underlying interfaces without invoking the OAL can only reach peers located on the same OMNI link segment. However, an ANET proxy that receives the original IP packet can forward it further by performing OAL encapsulation with source set to its own address and destination set to the OAL destination corresponding to the final destination (i.e., even if the OAL destination is on a different OMNI link segment).

Original IP packets sent directly over underlying interfaces are subject to the same path MTU related issues as for any Internetworking path, and do not include per-packet identifications that can be used for data origin verification and/or link-layer retransmissions. Original IP packets presented directly to an underlying interface that exceed the underlying network path MTU are
dropped with an ordinary ICMPv6 Packet Too Big (PTB) message returned. These PTB messages are subject to loss [RFC2923] the same as for any non-OMNI IP interface.

The OMNI interface encapsulation/decapsulation layering possibilities are shown in Figure 2 below. In the figure, imaginary vertical lines drawn between the Network Layer and Underlying interfaces denote the encapsulation/decapsulation layering combinations possible. Common combinations include NULL (i.e., direct access to underlying interfaces with or without using the OMNI interface), OMNI/IP, OMNI/UDP/IP, OMNI/UDP/IP/L2, OMNI/OAL/UDP/IP, OMNI/OAL/UDP/L2, etc.

![Figure 2: OMNI Interface Layering](image)

The OMNI/OAL model gives rise to a number of opportunities:

- MNs receive a MNP from the MS, and coordinate 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 8 and assigns the LLA to the OMNI interface. Since MNP-LLAs are uniquely derived from an MNP, no Duplicate Address Detection (DAD) or Multicast Listener Discovery (MLD) messaging is necessary.

- since Temporary ULAs are statistically unique, they can be used without DAD, e.g. for MN-to-MN communications until an MNP-LLA is obtained.

- underlying 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.

- 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.

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

- 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.

- 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.

- The OAL supports lossless and adaptive path MTU mitigations not available for communications directly over the underlying interfaces themselves. The OAL supports "packing" of multiple IP payload packets within a single OAL packet.

- The OAL applies per-packet identification values that allow for link-layer reliability and data origin authentication.

- 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.

- 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 realized when the architectural layering is bypassed in this way.

Figure 3 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.
Figure 3: 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 original IP 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.
5. OMNI Interface Maximum Transmission Unit (MTU)

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 as discussed in Section 4, where the interfaces (and their associated *NET paths) may have diverse MTUs. OMNI interface considerations for accommodating original IP packets of various sizes are discussed in the following sections.

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 original IP packets of at least 1280 bytes without generating an IPv6 Path MTU Discovery (PMTUD) Packet Too Big (PTB) message [RFC8201]. (While the source can apply "source fragmentation" for locally-generated IPv6 packets up to 1500 bytes and larger still if it knows the destination configures a larger MRU, 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 OMNI interface therefore MUST set DF to 0 in the IPv4 encapsulation headers of carrier packets that are no larger than 576 bytes, and SHOULD set DF to 1 in larger carrier packets. (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 6.9.)

The OMNI interface configures an MTU and MRU of 9180 bytes [RFC2492]; the size is therefore not a reflection of the underlying interface or *NET path MTUs, but rather determines the largest original IP packet the OAL (and/or underlying interface) can forward or reassemble. For each OAL destination (i.e., for each OMNI link neighbor), the OAL source may discover "hard" or "soft" Reassembly Limit values smaller than the MRU based on receipt of IPv6 ND messages with OMNI Reassembly Limit sub-options (see: Section 12.1.11). The OMNI interface employs the OAL as an encapsulation sublayer service to transform original IP packets into OAL packets/fragments, and the OAL
The OMNI Adaptation Layer (OAL)

When an OMNI interface forwards an original IP packet from the network layer for transmission over one or more underlying interfaces, the OMNI Adaptation Layer (OAL) acting as the OAL source drops the packet and returns a PTB message if the packet exceeds the MRU and/or the hard Reassembly Limit for the intended OAL destination. Otherwise, the OAL source applies encapsulation to form OAL packets and fragmentation to produce resulting OAL fragments suitable for *NET encapsulation and transmission as carrier packets over underlying interfaces as described in Section 6.1.

These carrier packets travel over one or more underlying networks bridged by OAL intermediate nodes, which re-encapsulate by removing the *NET headers of the first underlying network and appending *NET headers appropriate for the next underlying network in succession. After re-encapsulation by zero or more OAL intermediate nodes, the carrier packets arrive at the OAL destination.

When the OAL destination receives the carrier packets, it discards the *NET headers and reassembles the resulting OAL fragments into an OAL packet as described in Section 6.3. The OAL destination then decapsulates the OAL packet to obtain the original IP packet, which it then delivers to the network layer.

Detailed operations of the OAL are discussed in the following sections.

6.1. OAL Source Encapsulation and Fragmentation

When the network layer forwards an original IP packet into the OMNI interface, the OAL source inserts an IPv6 encapsulation header but does not decrement the Hop Limit/TTL of the original IP packet since encapsulation occurs at a layer below IP forwarding [RFC2473]. The OAL source copies the "Type of Service/Traffic Class" [RFC2983], "Flow Label" [RFC6438] (for IPv6) and "Congestion Experienced" [RFC3168] values in the original packet’s IP header into the corresponding fields in the OAL header. The OAL source finally sets the OAL header IPv6 Hop Limit to a small value (e.g., 16) large enough to allow forwarding over a small number of OMNI link segments and sets the Payload Length to the length of the original IP packet.

The OAL next selects source and destination addresses for the IPv6 header of the resulting OAL packet. MN OMNI interfaces set the OAL IPv6 header source address to a Unique Local Address (ULA) based on
the Mobile Network Prefix (MNP-ULA), while AR and MSE OMNI interfaces set the source address to an Administrative ULA (ADM-ULA) (see: Section 9). When a MN OMNI interface does not (yet) have an MNP-ULA, it can use a Temporary ULA and/or Host Identity Tag (HIT) instead (see: Section 22).

When the OAL source forwards an original IP packet toward a final destination via an ANET underlying interface, it sets the OAL IPv6 header source address to its own ULA and sets the destination 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 source then fragments the OAL packet if necessary, encapsulates the OAL fragments in any ANET headers and sends the resulting carrier packets to the ANET peer which either reassembles before forwarding if the OAL destination is its own ULA or forwards the fragments toward the true OAL destination without first reassembling otherwise.

When the OAL source forwards an original IP packet toward a final destination via an INET underlying interface, it sets the OAL IPv6 header source address to its own ULA and sets the destination to the ULA of an OAL destination node on the final *NET segment. The OAL source then fragments the OAL packet if necessary, encapsulates the OAL fragments in any *NET headers and sends the resulting carrier packets toward the OAL destination on the final segment OMNI node which reassembles before forwarding the original IP packets toward the final destination.

Following OAL IPv6 encapsulation and address selection, the OAL source next appends a 2 octet trailing Checksum (initialized to 0) at the end of the original IP packet while incrementing the OAL header IPv6 Payload Length field to reflect the addition of the trailer. The format of the resulting OAL packet following encapsulation is shown in Figure 4:

```
+----------+-----+-----+-----+-----+-----+-----+----+
|  OAL Hdr |         Original IP packet        |Csum|
+----------+-----+-----+-----+-----+-----+-----+----+
```

Figure 4: OAL Packet Before Fragmentation

The OAL source next selects a 32-bit Identification value for the packet, beginning with an unpredictable value for the initial OAL packet per [RFC7739] and monotonically incrementing for each successive OAL packet until a new initial value is chosen.

The OAL source then calculates the 2’s complement (mod 256) Fletcher’s checksum [CKSUM][RFC2328][RFC0905] over the entire OAL
packet beginning with a pseudo-header of the IPv6 header similar to that found in Section 8.1 of [RFC8200]. The OAL IPv6 pseudo-header is formed as shown in Figure 5:

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

Figure 5: OAL IPv6 Pseudo-Header

The OAL source then inserts a single OMNI Routing Header (ORH) if necessary (see: [I-D.templin-intarea-6706bis]) while incrementing Payload Length to reflect the addition of the ORH (note that the late addition of the ORH is not covered by the trailing checksum).

The OAL source next fragments the OAL packet if necessary while assuming the IPv4 minimum path MTU (i.e., 576 bytes) as the worst case for OAL fragmentation regardless of the underlying interface IP protocol version since IPv6/IPv4 protocol translation and/or IPv6-in-IPv4 encapsulation may occur in any *NET path. By always assuming the IPv4 minimum even for IPv6 underlying interfaces, the OAL source may produce smaller fragments with additional encapsulation overhead but will always interoperate and never run the risk of loss due to an MTU restriction or due to presenting an underlying interface with a carrier packet that exceeds its MRU. Additionally, the OAL path could traverse multiple *NET "segments" with intermediate OAL forwarding nodes performing re-encapsulation where the *NET encapsulation of the previous segment is replaced by the *NET
encapsulation of the next segment which may be based on a different IP protocol version and/or encapsulation sizes.

The OAL source therefore assumes a default minimum path MTU of 576 bytes at each *NET segment for the purpose of generating OAL fragments for *NET encapsulation and transmission as carrier packets. In the worst case, each successive *NET segment may re-encapsulate with either a 20 byte IPv4 or 40 byte IPv6 header, an 8 byte UDP header and in some cases an IP security encapsulation (40 bytes maximum assumed). Any *NET segment may also insert a maximum-length (40 byte) ORH as an extension to the existing 40 byte OAL IPv6 header plus 8 byte Fragment Header if an ORH was not already present. Assuming therefore an absolute worst case of (40 + 40 + 8) = 88 bytes for *NET encapsulation plus (40 + 40 + 8) = 88 bytes for OAL encapsulation leaves (576 - 88 - 88) = 400 bytes to accommodate a portion of the original IP packet/fragment. The OAL source therefore sets a minimum Maximum Payload Size (MPS) of 400 bytes as the basis for the minimum-sized OAL fragment that can be assured of traversing all segments without loss due to an MTU/MRU restriction. The Maximum Fragment Size (MFS) for OAL fragmentation is therefore determined by the MPS plus the size of the OAL encapsulation headers. (Note that the OAL source includes the 2 octet trailer as part of the payload during fragmentation, and the OAL destination regards it as ordinary payload until reassembly and checksum verification are complete.)

The OAL source SHOULD maintain "path MPS" values for individual OAL destinations initialized to the minimum MPS and increased to larger values (up to the OMNI interface MTU) if better information is known or discovered. For example, when *NET peers share a common underlying link or a fixed path with a known larger MTU, the OAL source can base path MPS on this larger size (i.e., instead of 576 bytes) as long as the *NET peer reassembles before re-encapsulating and forwarding (while re-fragmenting if necessary). Also, if the OAL source has a way of knowing the maximum *NET encapsulation size for all segments along the path it may be able to increase path MPS to reserve additional room for payload data. The OAL source must include the uncompressed OAL header size in its path MPS calculation, since a full header could be included at any time.

The OAL source can also actively probe individual OAL destinations to discover larger path MPS values using packetization layer probes per [RFC4821][RFC8899], but care must be taken to avoid setting static values for dynamically changing paths leading to black holes. The probe involves sending an OAL packet larger than the current path MPS and receiving a small acknowledgement message in response (with the possible receipt of link-layer error message in case the probe was lost). For this purpose, the OAL source can send an NS message with one or more OMNI options with large PadN sub-options (see:
Section 12) in order to receive a small NA response from the OAL destination. While observing the minimum MPS will always result in robust and secure behavior, the OAL source should optimize path MPS values when more efficient utilization may result in better performance (e.g. for wireless aviation data links).

When the OAL source performs fragmentation, it SHOULD produce the minimum number of non-overlapping fragments under current MPS constraints, where each non-final fragment MUST be of equal length at least as large as the minimum MPS, while the final fragment MAY be of different length. The OAL source also converts all original IP packets no larger than the current MPS into "atomic fragments" by including a Fragment Header with Fragment Offset and More Fragments both set to 0. The OAL source finally encapsulates the fragments in *NET headers to form carrier packets and forwards them over an underlying interface, while retaining the fragments and their ordinal positions (i.e., as Frag #0, Frag #1, Frag #2, etc.) for a timeout period in case link-layer retransmission is requested. The formats of OAL fragments and carrier packets are shown in Figure 6.
6.2. OAL *NET Encapsulation and Re-Encapsulation

During *NET encapsulation, OAL sources first encapsulate each OAL fragment in a UDP header as the first *NET encapsulation sublayer if NAT traversal, packet filtering middlebox traversal and/or OAL header compression are necessary. The OAL then optionally appends additional encapsulation sublayer headers, then presents the *NET packet to an underlying interface. This layering can be seen in Figure 2.

When a UDP header is included, the OAL source next sets the UDP source port to a constant value that it will use in each successive carrier packet it sends to the next OAL hop. For packets sent to an MSE, the OAL source sets the UDP destination port to 8060, i.e., the IANA-registered port number for AERO. For packets sent to a MN peer, the source sets the UDP destination port to the cached port value for this peer. The OAL source then sets the UDP length to the total length of the OAL fragment in correspondence with the OAL header.
Payload Length (i.e., the UDP length and IPv6 Payload Length must agree). The OAL source finally sets the UDP checksum to 0 [RFC6935][RFC6936] since the only fields not already covered by the OAL checksum or underlying *NET CRCs are the Fragment Header fields, and any corruption in those fields will be garbage collected by the reassembly algorithm. The UDP encapsulation header is often used in association with IP encapsulation, but may also be used between neighbors on a shared physical link with a true L2 header format such as for transmission over IEEE 802 Ethernet links. This document therefore requests a new Ether Type code assignment TBD1 in the IANA 'ieee-802-numbers' registry for direct User Datagram Protocol (UDP) encapsulation over IEEE 802 Ethernet links (see: Section 25).

For *NET encapsulations, the OAL source next copies the "Type of Service/Traffic Class" [RFC2983], "Congestion Experienced" [RFC3168] and "Flow Label" [RFC6438] (for IPv6) values in the OAL IPv6 header into the corresponding fields in the *NET IP header. For carrier packets undergoing re-encapsulation, OAL intermediate nodes instead copy these values from the previous hop *NET encapsulation header into both the OAL IPv6 header and the next hop *NET encapsulation header, i.e., the IP values are transferred between *NET encapsulation headers and *not* copied from the OAL header. During re-encapsulation, the intermediate node decrements the OAL IPv6 header Hop Limit and discards the carrier packet if the value reaches 0.

Following *NET encapsulation/re-encapsulation, the OAL source sends the resulting carrier packets over one or more underlying interfaces. The 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 the underlying interface layer (e.g., as for a tunnel virtual interface) such that carrier 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.

6.3. OAL Destination Decapsulation and Reassembly

When an OMNI interface receives a carrier packet from an underlying interface, the OAL destination discards the *NET encapsulation headers and examines the OAL header of the enclosed OAL fragment. If
the OAL fragment is addressed to a different node, the OAL destination re-encapsulates and forwards as discussed below. If the OAL fragment is addressed to itself, the OAL destination creates or updates a checklist for this (Source, Destination, Identification)-tuple to track the fragments already received (i.e., by examining the Payload Length, Fragment Offset, More Fragments and Identification values supplied by the OAL source). The OAL destination verifies that all non-final OAL fragments are of equal length no less than the minimum MPS and that no fragments overlap or leave "holes", while dropping any non-conforming fragments. The OAL destination records each conforming OAL fragment’s ordinal position based on the OAL header Payload Length and Fragment Offset values (i.e., as Frag #0, Frag #1, Frag #2, etc.) and admits each fragment into the reassembly cache.

When reassembly is complete, the OAL destination removes the ORH if present while decrementing Payload Length to reflect the removal of the ORH. The OAL destination next verifies the resulting OAL packet’s checksum and discards the packet if the checksum is incorrect. If the OAL packet was accepted, the OAL destination then removes the OAL header/trailer, then delivers the original IP packet to the network layer. Note that link layers include a CRC-32 integrity check which provides effective hop-by-hop error detection in the underlying network for payload sizes up to the OMNI interface MTU [CRC], but that some hops may traverse intermediate layers such as tunnels over IPv4 that do not include integrity checks. The trailing Fletcher checksum therefore allows the OAL destination to detect OAL packet splicing errors due to reassembly misassociations and/or to verify integrity for OAL packets whose fragments may have traversed unprotected underlying network hops [CKSUM]. The Fletcher algorithm also provides diversity with respect to both lower layer CRCs and upper layer Internet checksums as part of a complimentary multi-layer integrity assurance architecture.

6.4. OAL Header Compression

When the OAL source and destination are on the same *NET segment, no ORH is needed and carrier packet header compression is possible. When the OAL source and destination exchange initial IPv6 ND messages as discussed in the following Sections, each caches the observed *NET UDP source port and source IP (or L2) address associated with the OAL IPv6 source address found in the full-length OAL IPv6 header. After the initial IPv6 ND message exchange, the OAL source can begin applying OAL Header Compression to significantly reduce the encapsulation overhead required in each carrier packet.

When the OAL source determines that header compression state has been established (i.e., following the IPv6 ND message exchange), it can
begin sending OAL fragments with significant portions of the IPv6 header and Fragment Header omitted thereby reducing the amount of encapsulation overhead. For OAL first-fragments (including atomic fragments), the OMNI Compressed Header - Type 0 (OCH-0) is used and formatted as shown in Figure 7:

```
<table>
<thead>
<tr>
<th>Version</th>
<th>Traffic Class</th>
<th>Traffic Class</th>
<th>Flow Label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Source port</td>
<td>Destination port</td>
<td>U</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>---------------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>Checksum</td>
<td>P</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>---------------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td>Identification (0 -1)</td>
<td>M</td>
<td>Identification (2-3)</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>---------------</td>
<td>------------</td>
</tr>
</tbody>
</table>
```

Figure 7: OMNI Compressed Header - Type 0 (OCH-0)

In this format, the UDP header appears in its entirety in the first 8 octets, then followed by the first 4 octets of the IPv6 header with the remainder omitted. (The IPv6 Version field is set to the value 0 to distinguish this header from a true IP protocol version number and from OCH-1 - see below.) The compressed IPv6 header is then followed by a compressed IPv6 Fragment Header with the Fragment Offset field and two Reserved bits omitted (since these fields always encode the value 0 in first-fragments), and with the More Fragments (M) bit relocated to the least significant bit of the first Reserved field. The OCH-0 header is then followed by the OAL fragment body, and the UDP length field is reduced by 38 octets (i.e., the difference in length between full-length IPv6 and Fragment Headers and the length of the compressed headers).

For OAL non-first fragments (i.e., those with non-zero Fragment Offsets), the OMNI Compressed Header - Type 1 (OCH-1) is used and formatted as shown in Figure 8:
In this format, the UDP header appears in its entirety in the first 8 octets, but all IPv6 header fields except for the most significant Version (V) bit are omitted. (The V bit is set to the value 1 to distinguish this header from a true IP protocol version number and from OCH-0.) The V bit is followed by a single Reserved (R) bit and the More Fragments (M) bit in a compressed IPv6 Fragment Header with the Next Header and first Reserved fields omitted. The OCH-1 header is then followed by the OAL fragment body, and the UDP length field is reduced by 42 octets (i.e., the difference in length between full-length IPv6 and Fragment Headers and the length of the compressed headers).

When the OAL destination receives a carrier packet with an OCH, it first determines the OAL IPv6 source and destination addresses by examining the UDP source port and L2 source address, then determines the length by examining the UDP length. The OAL destination then examines the (V)ersion field immediately following the UDP header. If the (4-bit) Version field encodes the value 0, the OAL destination processes the remainder of the header as an OCH-0, then reconstitutes the full-sized IPv6 and Fragment Headers and adds this OAL fragment to the reassembly buffer if necessary. If the (1-bit) V bit encodes the value 1, the OAL destination instead processes the remainder of the header as an OCH-1, then reconstitutes the full-sized IPv6 and Fragment Headers and adds this OAL fragment to the reassembly buffer. Note that, since the OCH-1 does not include Traffic Class, Flow Label or Next Header information, the OAL destination writes the value 0 into these fields when it reconstitutes the full headers. These values will be correctly populated during reassembly after an OAL first fragment with an OCH-0 or uncompressed OAL header arrives.
6.5. OAL Fragment Identification Window Maintenance

As noted above, the OAL source establishes a window of 32-bit Identifications beginning with an unpredictable value for the initial message [RFC7739] and monotonically incrementing for each successive OAL packet until a new initial value is chosen. The OAL source asserts the starting value by including it as the Identification in an IPv6 ND NS/RS messages. When the OAL destination receives the IPv6 ND message, it resets the Identification window for this OAL source to the new value coded in the message’s OAL header and expects future OAL fragments received from this OAL source to include sequential Identification values (subject to loss and reordering) until the neighbor reachable time expires or the OAL source sends a new IPv6 ND message.

For example, if the OAL destination receives an NS/RS message with Identification 0x12345678, it resets the window for this OAL source to begin with 0x12345678 and examines the Identification values in subsequent OAL fragments received from this OAL source. If the Identification values of subsequent OAL fragments fall within the window of (0x12345678 + N) the OAL destination accepts the fragment; otherwise, it silently drops the fragment (where "N" represents the maximum number of fragments expected before the neighbor reachable time expires).

While monitoring the current window, the OAL destination must accept new NS/RS Identification values even if outside the current window. The new Identification value resets the OAL destination’s window start, and the window processing continues from this new starting point while allowing a period of overlap in case OAL fragments with Identification values from a previous window are still in flight. Note also that unsolicited NA messages must include Identification values within the current window, and therefore do not reset the current window.

This implies that an IPv6 ND message used to reset the Identification window should fit within a single OAL fragment (i.e., within current MPS constraints), since a fragmented IPv6 ND message with an out-of-window Identification value could be part of a DoS attack. While larger IPv6 ND messages (up to the OMNI interface MTU) can certainly be subject to OAL fragmentation, their Identification should be within the current window maintained by the OAL destination to increase the likelihood that they will be accepted.
6.6. OAL Fragment Retransmission

When the OAL source sends carrier packets with OAL fragments to an OAL destination, the source caches them for a timeout period in case retransmission may be necessary. (The timeout duration is an implementation matter, and may be influenced by factors such as packet arrival rates, OAL source/destination round trip times, etc.) The OAL destination in turn maintains a checklist for the (Source, Destination, Identification)-tuple of each new OAL fragment received and notes the ordinal positions of fragments already received (i.e., as Frag #0, Frag #1, Frag #2, etc.).

If the OAL destination notices some OAL fragments missing after most other fragments within the same Identification window have already arrived, it may send an IPv6 ND unsolicited Neighbor Advertisement (uNA) message to the OAL source that originated the fragments to report loss. The OAL destination creates a uNA message with an OMNI option containing an authentication sub-option to provide authentication (if the OAL source is on an open Internetwork) followed by a Fragmentation Report sub-option that includes a list of (Identification, Bitmap)-tuples for OAL fragments received and missing from this OAL source (see: Section 12). The OAL destination signs the message if an authentication sub-option is included, performs OAL encapsulation (with the its own address as the OAL source and the source address of the message that prompted the uNA as the OAL destination) and sends the message to the OAL source.

When the OAL source receives the uNA message, it authenticates the message using authentication sub-option (if present) then examines the Fragmentation Report. For each (Source, Destination, Identification)-tuple, the OAL source determines whether it still holds the original OAL fragments in its cache and retransmits any for which the Bitmap indicated a loss event. For example, if the Bitmap indicates that the ordinal OAL fragments Frag #3, Frag #7, Frag #10 and Frag #13 from the same OAL packet are missing the OAL source retransmits these fragments only and no others.

Note that the goal of this service is to provide a light-weight link-layer Automatic Repeat Request (ARQ) capability in the spirit of Section 8.1 of [RFC3819]. Rather than provide true end-to-end reliability, however, the service provides timely link-layer retransmissions that may improve packet delivery ratios and avoid some delays inherent in true end-to-end services.
6.7. OAL MTU Feedback Messaging

When the OMNI interface forwards original IP packets from the network layer, it invokes the OAL and returns internally-generated 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.

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. In particular, the OAL source drops the packet and returns a PTB hard error if the packet exceeds the OAL destination MRU. However, the OMNI interface can also forward large original IP packets via OAL encapsulation and fragmentation while at the same time returning PTB soft error messages (subject to rate limiting) if it deems the original IP 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 OMNI interface can therefore continuously forward packets without loss while returning PTB soft error messages recommending a smaller size if necessary. 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), but can soon resume sending larger packets if the soft errors subside.

An OAL source sends PTB soft error messages by setting the ICMPv4 header "unused" field or ICMPv6 header Code field to the value 1 if a original IP packet was deemed lost (e.g., due to reassembly timeout) or to the value 2 otherwise. The OAL source sets the PTB destination address to the original IP packet source, and sets the source address to one of its OMNI interface unicast/anycast addresses that is routable from the perspective of the original source. The OAL source then sets the MTU field to a value smaller than the original packet size but no smaller than 576 for ICMPv4 or 1280 for ICMPv6, writes the leading portion of the original IP packet into the "packet in error" field, and returns the PTB soft error to the original source. When the original source receives the PTB soft error, it temporarily reduces the size of the packets it sends the same as for hard errors but may seek to increase future packet sizes dynamically while no further soft errors are arriving. (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, packetization layer
probing strategies [RFC4821][RFC8899] must be aware that PTB hard or
soft errors may arrive at any time, i.e., even following a successful
probe (this is the same consideration as for an ordinary path
fluctuation following a successful probe).

An OAL destination may experience reassembly cache congestion, and
can return uNA messages to the OAL source that originated the
fragments (subject to rate limiting) to advertise reduced hard/soft
Reassembly Limits and/or to report individual reassembly failures.
The OAL destination creates a uNA message with an OMNI option
containing an authentication message sub-option (if the OAL source is
on an open Internetwork) followed optionally by at most one hard and
one soft Reassembly Limit sub-options with reduced hard/soft values,
and with one of them optionally including the leading portion an OAL
first fragment containing the header of an original IP packet whose
source must be notified (see: Section 12). The OAL destination
encapsulates as much of the OAL first fragment (beginning with the
OAL header) as will fit in the "OAL First Fragment" field of sub-
option without causing the entire uNA message to exceed the minimum
MPS, signs the message if an authentication sub-option is included,
performs OAL encapsulation (with the its own address as the OAL
source and the source address of the message that prompted the uNA as
the OAL destination) and sends the message to the OAL source.

When the OAL source receives the uNA message, it records the new
hard/soft Reassembly Limit values for this OAL destination if the
OMNI option includes Reassembly Limit sub-options. If a hard or soft
Reassembly Limit sub-option includes an OAL First Fragment, the OAL
source next sends a corresponding network layer PTB hard or soft
error to the original source to recommend a smaller size. For hard
errors, the OAL source sets the PTB Code field to 0. For soft
errors, the OAL source sets the PTB Code field to 1 if the L flag in
the Reassembly Limit sub-option is 1; otherwise, the OAL source sets
the Code field to 2. The OAL source crafts the PTB by extracting the
leading portion of the original IP packet from the OAL First Fragment
field (i.e., not including the OAL header) and writes it in the
"packet in error" field of a PTB with destination set to the original
IP packet source and source set to one of its OMNI interface unicast/
anycast addresses that is routable from the perspective of the
original source. For future transmissions, if the original IP packet
is larger than the hard Reassembly Limit for this OAL destination the
OAL source drops the packet and returns a PTB hard error with MTU set
to the hard Reassembly Limit. If the packet is no larger than the
current hard Reassembly Limit but larger than the current soft limit,
the OAL source can also return PTB soft errors (subject to rate
limiting) with Code set to 2 and MTU set to the current soft limit
while still forwarding the packet to the OMNI destination.
Original sources that receive PTB soft errors can dynamically tune the size of the original IP 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. The OMNI interface 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 a hard limit, original sources that receive soft errors can begin sending larger packets without waiting for the recommended 10 minutes specified for PTB hard errors [RFC1191][RFC8201]. The OMNI interface therefore provides an adaptive service that accommodates MTU diversity especially well-suited for dynamic multilink environments.

6.8. OAL Requirements

In light of the above, OAL sources, destinations and intermediate nodes observe the following normative requirements:

- OAL sources MUST NOT send OAL fragments including original IP packets larger than the OMNI interface MTU or the OAL destination hard Reassembly Limit, i.e., whether or not fragmentation is needed.
- OAL sources MUST NOT perform OAL fragmentation for original IP packets smaller than the minimum MPS minus the trailer size, and MUST produce non-final fragments that contain equal-length payloads no smaller than the minimum MPS when performing fragmentation.
- OAL sources MUST NOT send OAL fragments that include any extension headers other than a single ORH and a single Fragment Header.
- OAL intermediate nodes SHOULD and OAL destinations MUST unconditionally drop OAL packets/fragments including original IP packets larger than the OMNI interface MRU and/or OAL destination Reassembly Limit, i.e., whether or not reassembly was needed.
- OAL intermediate nodes SHOULD and OAL destinations MUST unconditionally drop any non-final OAL fragments containing a payload smaller than the minimum MPS.
- OAL intermediate nodes SHOULD and OAL destinations MUST unconditionally drop OAL fragments that include any extension headers other than a single ORH and a single Fragment Header.
o OAL destination nodes MUST drop any new OAL non-final fragments of
different length than other non-final fragments that have already
been received, and MUST drop any new OAL fragments with Offset and
Payload length that would overlap with other fragments and/or
leave too-small holes between fragments that have already been
received.

Note: Under the minimum MPS, ordinary 1500 byte original IP packets
would require at most 4 OAL fragments, with each non-final fragment
containing 400 payload bytes and the final fragment containing 302
payload bytes (i.e., the final 300 bytes of the original IP packet
plus the 2 octet trailer). Likewise, maximum-length 9180 byte
original IP packets would require at most 23 fragments. For all
packet sizes, the likelihood of successful reassembly may improve
when the OMNI interface sends all fragments of the same fragmented
OAL packet consecutively over the same underlying interface.
Finally, an assured minimum/path MPS allows continuous operation over
all paths including those that traverse bridged L2 media with
dissimilar MTUs.

Note: Certain legacy network hardware of the past millennium was
unable to accept packet "bursts" resulting from an IP fragmentation
event - even to the point that the hardware would reset itself when
presented with a burst. This does not seem to be a common problem in
the modern era, where fragmentation and reassembly can be readily
demonstrated at line rate (e.g., using tools such as 'iperf3') even
over fast links on average hardware platforms. Even so, the OAL
source could impose an inter-fragment delay while the OAL destination
is reporting reassembly congestion (see: Section 6.7) and decrease
the delay when reassembly congestion subsides.

6.9. OAL 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]. Additionally, inclusion of the OAL checksum would make it very difficult for an attacker who could somehow predict a fragment identification value to inject malicious fragments resulting in undetected reassemblies of bad data.

4. Evasion of Network Intrusion Detection Systems (NIDS) - this threat is mitigated by setting a minimum MPS for OAL fragmentation, which defeats all "tiny fragment"-based attacks.

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 carrier packets while the fragments of old packets using the same ID are still alive in the network [RFC4963]. However, since the largest carrier packet that will be sent via an IPv4 path with DF = 0 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.

Finally, [RFC6980] documents fragmentation security concerns for large IPv6 ND messages. These concerns are addressed when the OMNI interface employs the OAL instead of directly fragmenting the IPv6 ND message itself. For this reason, OMNI interfaces MUST NOT send IPv6 ND messages larger than the OMNI interface MTU, and MUST employ OAL encapsulation and fragmentation for IPv6 ND messages larger than the current MPS for this OAL destination.

6.10. OAL Super-Packets

By default, the OAL source includes a 40-byte IPv6 encapsulation header for each original IP packet during OAL encapsulation. The OAL source also calculates and appends a 2 octet trailing Fletcher checksum then performs fragmentation such that a copy of the 40-byte IPv6 header plus an 8-byte IPv6 Fragment Header is included in each OAL fragment (when an ORH is added, the OAL encapsulation headers become larger still). However, these encapsulations may represent excessive overhead in some environments. OAL header compression can dramatically reduce the amount of encapsulation overhead, however a complimentary technique known as "packing" (see:
When the OAL source has multiple original IP packets to send to the same OAL destination with total length no larger than the OAL destination MRU, it can concatenate them into a super-packet encapsulated in a single OAL header and trailing checksum. Within the OAL super-packet, the IP header of the first original IP packet (iHa) followed by its data (iDa) is concatenated immediately following the OAL header, then the IP header of the next original packet (iHb) followed by its data (iDb) is concatenated immediately following the first original packet, etc. with the trailing checksum included last. The OAL super-packet format is transposed from [I-D.ietf-intarea-tunnels] and shown in Figure 9:

<-------- Original IP packets -------->
+----------+
| iHa | iDa |
+----------+

+----------+
| iHb | iDb |
+----------+

+----------+
| iHc | iDc |
+----------+

+----------+----------+----------+----------+----------+----------+----------+----------+
| OAL Hdr | iHa | iDa | iHb | iDb | iHc | iDc | Csum |
+----------+----------+----------+----------+----------+----------+----------+----------+

<--- OAL "Super-Packet" with single OAL Hdr/Csum --->

Figure 9: OAL Super-Packet Format

When the OAL source prepares a super-packet, it applies OAL fragmentation and *NET encapsulation then sends the carrier packets to the OAL destination. When the OAL destination receives the super-packet it reassembles if necessary, verifies and removes the trailing checksum, then regards the remaining OAL header Payload Length as the sum of the lengths of all payload packets. The OAL destination then selectively extracts each original IP packet (e.g., by setting pointers into the super-packet buffer and maintaining a reference count, by copying each packet into a separate buffer, etc.) and forwards each packet to the network layer. During extraction, the OAL determines the IP protocol version of each successive original IP
packet ‘j’ by examining the four most-significant bits of iH(j), and determines the length of the packet by examining the rest of iH(j) according to the IP protocol version.

Note that OMNI interfaces must take care to avoid processing super-packet payload elements that would subvert security. Specifically, if a super-packet contains a mix of data and control payload packets (which could include critical security codes), the node MUST NOT process the data packets before processing the control packets.

7. Frame Format

The OMNI interface forwards original IP packets from the network layer by first invoking the OAL to create OAL packets/fragments if necessary, then including any *NET encapsulations and finally engaging the native frame format of the 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 various forms of tunnels the frame format is found in the appropriate tunneling specification, etc.

See Figure 2 for a map of the various *NET layering combinations possible. For any layering combination, the final layer (e.g., UDP, IP, Ethernet, etc.) must have an assigned number and frame format representation that is compatible with the selected underlying interface.

8. 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. Non-MNP routes are also represented the same as for MNP-LLAs, but include a GUA prefix that is not properly covered by the MSP.

- 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).

- 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 0xffffffff, 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 0xffffffff through 0xffffffff is reserved for future use.

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 different MNP- and ADM-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].

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 foreseeable future.

Note: Even though this document honors the 64-bit boundary in IPv6 addressing, it specifies 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. Modern routing protocol implementations honor IPv6 prefixes of all lengths, up to and including /128.
9. Unique-Local Addresses (ULAs)

OMNI domains use IPv6 Unique-Local Addresses (ULAs) as the source and destination addresses in OAL packet IPv6 encapsulation headers. ULAs are only routable within the scope of 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]. All nodes in the same OMNI domain configure the same 40-bit Global ID as the OMNI domain identifier. The statistic uniqueness of the 40-bit pseudo-random Global ID allows different OMNI domains to be joined together in the future without requiring renumbering.

Each OMNI link instance is identified by a value between 0x0000 and 0x0fff in bits 48-63 of [ULA]::/48; the values 0xff00 through 0xfffe are reserved for future use, and the value 0xffff denotes the presence of a Temporary ULA (see below). 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:

- 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).

- 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).

- the ADM-ULA corresponding to fe80::1000/112 is simply [ULA]:1010::1000/112.

- the ADM-ULA corresponding to fe80::/128 is simply [ULA]:1010::/128.

- 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 underlying 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 14).

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 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].

Temporary ULAs are constructed per [RFC8981] based on the prefix [ULA]:ffff::/64 and used by MNs when they have no other addresses. Temporary ULAs can be used for MN-to-MN communications outside the context of any supporting OMNI link infrastructure, and can also be used as an initial address while the MN is in the process of procuring an MNP. Temporary ULAs are not routable within the OMNI routing system, and are therefore useful only for OMNI link "edge" communications. Temporary ULAs employ optimistic DAD principles [RFC4429] since they are probabilistically unique.

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 10).

10. 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). Fixed correspondent node networks reachable from the OMNI domain are represented by non-MNP GUA prefixes that are not derived from the MSP, but are treated in all other ways the same as for MNPs.

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.

11. Node Identification

OMNI MNs and MSEs that connect over open Internetworks include a unique node identification value for themselves in the OMNI options of their IPv6 ND messages (see: Section 12.1.13). One useful identification value alternative is the Host Identity Tag (HIT) as specified in [RFC7401], while Hierarchical HITs (HHITs) [I-D.ietf-drip-rid] may provide a better alternative in certain domains such as the Unmanned (Air) Traffic Management (UTM) service for Unmanned Air Systems (UAS). Another alternative is the Universally Unique IDentifier (UUID) [RFC4122] which can be self-generated by a node without supporting infrastructure with very low probability of collision.

When a MN is truly outside the context of any infrastructure, it may have no MNP information at all. In that case, the MN can use an IPv6 temporary ULA or (H)HIT as an IPv6 source/destination address for sustained communications in Vehicle-to-Vehicle (V2V) and (multihop) Vehicle-to-Infrastructure (V2I) scenarios. The MN can also propagate the ULA/(H)HIT into the multihop routing tables of (collective) Mobile/Vehicular Ad-hoc Networks (MANETs/VANETs) using only the vehicles themselves as communications relays.

When a MN connects to ARs over (non-multihop) protected-spectrum ANETs, an alternate form of node identification (e.g., MAC address, serial number, airframe identification value, VIN, etc.) may be sufficient. The MN can then include OMNI "Node Identification" sub-options (see: Section 12.1.13) in IPv6 ND messages should the need to transmit identification information over the network arise.

12. Address Mapping - Unicast

OMNI interfaces maintain a neighbor cache for tracking per-neighbor state and use the link-local address format specified in Section 8. 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 10:

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |     Length    |    Preflen    |  S/T-omIndex  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
˜                          Sub-Options                          ˜
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: OMNI Option Format

In this format:

- Type is set to TBD2.
- Length is set to the number of 8 octet blocks in the option. The value 0 is invalid, while the values 1 through 255 (i.e., 8 through 2040 octets, respectively) indicate the total length of the OMNI option.
- Preflen is an 8 bit field that determines the length of prefix associated with an LLA. Values 0 through 128 specify a valid prefix length (all other values are invalid). For IPv6 ND messages sent from a MN to the MS, Preflen applies to the IPv6 source LLA and provides the length that the MN is requesting or asserting to the MS. For IPv6 ND messages sent from the MS to the MN, Preflen applies to the IPv6 destination LLA and indicates the...
length that the MS is granting to the MN. For IPv6 ND messages sent between MS endpoints, Preflen provides the length associated with the source/target MN that is subject of the ND message.

- **S/T-omIndex** is an 8 bit field corresponds to the omIndex value for source or target underlying interface used to convey this IPv6 ND message. OMNI interfaces MUST number each distinct underlying interface with an omIndex value between ‘1’ and ‘255’ that represents a MN-specific 8-bit mapping for the actual ifIndex value assigned by network management [RFC2863] (the omIndex value ‘0’ is reserved for use by the MS). For RS and NS messages, S/T-omIndex corresponds to the source underlying interface the message originated from. For RA and NA messages, S/T-omIndex corresponds to the target underlying interface that the message is destined to. (For NS messages used for Neighbor Unreachability Detection (NUD), S/T-omIndex instead identifies the neighbor’s underlying interface to be used as the target interface to return the NA.)

- **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 12.1.

The OMNI option may appear in any IPv6 ND message type; it is processed by interfaces that recognize the option and ignored by all other interfaces. If multiple OMNI option instances appear in the same IPv6 ND message, the interface processes the Preflen and S/T-omIndex fields in the first instance and ignores those fields in all other instances. The interface processes the Sub-Options of all OMNI option instances in the same IPv6 ND message in the consecutive order in which they appear.

The OMNI option(s) in each IPv6 ND message may include full or partial information for the neighbor. The union of the information in the most recently received OMNI options is therefore retained, and the information is aged/removed in conjunction with the corresponding neighbor cache entry.

### 12.1. Sub-Options

Each 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:
Sub-Type is a 5-bit field that encodes the Sub-Option type. Sub-Options defined in this document are:

<table>
<thead>
<tr>
<th>Sub-Option Name</th>
<th>Sub-Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pad1</td>
<td>0</td>
</tr>
<tr>
<td>PadN</td>
<td>1</td>
</tr>
<tr>
<td>Interface Attributes (Type 1)</td>
<td>2</td>
</tr>
<tr>
<td>Interface Attributes (Type 2)</td>
<td>3</td>
</tr>
<tr>
<td>Traffic Selector</td>
<td>4</td>
</tr>
<tr>
<td>MS-Register</td>
<td>5</td>
</tr>
<tr>
<td>MS-Release</td>
<td>6</td>
</tr>
<tr>
<td>Geo Coordinates</td>
<td>7</td>
</tr>
<tr>
<td>DHCPv6 Message</td>
<td>8</td>
</tr>
<tr>
<td>HIP Message</td>
<td>9</td>
</tr>
<tr>
<td>Reassembly Limit</td>
<td>10</td>
</tr>
<tr>
<td>Fragmentation Report</td>
<td>11</td>
</tr>
<tr>
<td>Node Identification</td>
<td>12</td>
</tr>
<tr>
<td>Sub-Type Extension</td>
<td>30</td>
</tr>
</tbody>
</table>

Sub-Types 13-29 are available for future assignment for major protocol functions. Sub-Type 31 is reserved by IANA.

Sub-Length is an 11-bit field that encodes the length of the Sub-Option Data ranging from 0 to 2034 octets.

Sub-Option Data is a block of data with format determined by Sub-Type and length determined by Sub-Length.

During transmission, the OMNI interface codes Sub-Type and Sub-Length together in network byte order in 2 consecutive octets, where Sub-Option Data may be up to 2034 octets in length. This allows ample space for coding large objects (e.g., ASCII strings, domain names, protocol messages, security codes, etc.), while a single OMNI option is limited to 2040 octets the same as for any IPv6 ND option. If the Sub-Options to be coded would cause an OMNI option to exceed 2040 octets, the OMNI interface codes any remaining Sub-Options in additional OMNI option instances in the intended order of processing in the same IPv6 ND message. Implementations must therefore observe...
size limitations, and must refrain from sending IPv6 ND messages larger than the OMNI interface MTU. If the available OMNI information would cause a single IPv6 ND message to exceed the OMNI interface MTU, the OMNI interface codes as much as possible in a first IPv6 ND message and codes the remainder in additional IPv6 ND messages.

During reception, the OMNI interface processes each OMNI option Sub-Option while skipping over and ignoring any unrecognized Sub-Options. The OMNI interface processes the Sub-Options of all OMNI option instances in the consecutive order in which they appear in the IPv6 ND message, beginning with the first instance and continuing through any additional instances to the end of the message. If a Sub-Option length would cause processing to exceed the OMNI option total length, the OMNI interface accepts any Sub-Options already processed and ignores the final Sub-Option. The interface then processes any remaining OMNI options in the same fashion to the end of the IPv6 ND message.

Note: large objects that exceed the Sub-Option Data limit of 2034 octets are not supported under the current specification; if this proves to be limiting in practice, future specifications may define support for fragmenting large objects across multiple OMNI options within the same IPv6 ND message.

The following Sub-Option types and formats are defined in this document:

12.1.1. Pad1

```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
| S-Type=0|x|x|x|
+-+-+-+-+-+-+-+-+
```

Figure 13: Pad1

- Sub-Type is set to 0. If multiple instances appear in OMNI options of the same message all are processed.

- Sub-Type is followed by 3 ’x’ bits, set to any value on transmission (typically all-zeros) and ignored on receipt. Pad1 therefore consists of 1 octet with the most significant 5 bits set to 0, and with no Sub-Length or Sub-Option Data fields following.
12.1.2.  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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| S-Type=1 | Sub-length=N   | N padding octets ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

Figure 14: PadN

- Sub-Type is set to 1. If multiple instances appear in OMNI options of the same message all are processed.
- Sub-Length is set to N (from 0 to 2034) that encodes the number of padding octets that follow.
- Sub-Option Data consists of N octets, set to any value on transmission (typically all-zeros) and ignored on receipt.

12.1.3.  Interface Attributes (Type 1)

The Interface Attributes (Type 1) sub-option provides a basic set of attributes for underlying interfaces. Interface Attributes (Type 1) is deprecated throughout the rest of this specification, and Interface Attributes (Type 2) (see: Section 12.1.4) are indicated wherever the term "Interface Attributes" appears without an associated Type designation.

Nodes SHOULD NOT include Interface Attributes (Type 1) sub-options in IPv6 ND messages they send, and MUST ignore any in IPv6 ND messages they receive. If an Interface Attributes (Type 1) is included, it must have the following format:
Sub-Type is set to 2. If multiple instances with different omIndex values appear in OMNI option of the same message all are processed; if multiple instances with the same omIndex value appear, the first is processed and all others are ignored.

Sub-Length is set to N (from 4 to 2034) that encodes the number of Sub-Option Data octets that follow.

omIndex is a 1-octet field containing a value from 0 to 255 identifying the underlying interface for which the attributes apply.

omType is a 1-octet field containing a value from 0 to 255 corresponding to the underlying interface identified by omIndex.

Provider ID is a 1-octet field containing a value from 0 to 255 corresponding to the underlying interface identified by omIndex.

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").

Resvd is reserved for future use. Set to 0 on transmission and ignored on reception.

A 16-octet "Preferences" field immediately follows ‘Resvd’, with values P[00] through P[63] corresponding to the 64 Differentiated Service Code Point (DSCP) values [RFC2474]. 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.

12.1.4. Interface Attributes (Type 2)

The Interface Attributes (Type 2) sub-option provides L2 forwarding information for the multilink conceptual sending algorithm discussed in Section 14. The L2 information is used for selecting among potentially multiple candidate underlying interfaces that can be used to forward carrier packets to the neighbor based on factors such as DSCP preferences and link quality. Interface Attributes (Type 2) further includes link-layer address information to be used for either OAL encapsulation or direct UDP/IP encapsulation (when OAL encapsulation can be avoided).

Interface Attributes (Type 2) are the sole Interface Attributes format in this specification that all OMNI nodes must honor. Wherever the term "Interface Attributes" occurs throughout this specification without a "Type" designation, the format given below is indicated:

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| S-Type=3 |    Sub-length=N     |    omIndex    |    omType     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Provider ID  | Link  |R| API |   SRT   | FMT |   LHS (0 - 7) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               LHS (bits 8 - 31)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Bitmap(0)=0xff|P00|P01|P02|P03|P04|P05|P06|P07|P08|P09|P10|P11|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P28|P29|P30|P31| Bitmap(1)=0xff|P32|P33|P34|P35|P36| ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

---

Figure 16: Interface Attributes (Type 2)

- Sub-Type is set to 3. If multiple instances with different omIndex values appear in OMNI options of the same message all are processed; if multiple instances with the same omIndex value appear, the first is processed and all others are ignored.
Sub-Length is set to N (from 4 to 2034) that encodes the number of Sub-Option Data octets that follow. The 'omIndex', 'omType', 'Provider ID', 'Link', 'R' and 'API' fields are always present; hence, the remainder of the Sub-Option Data is limited to 2030 octets.

Sub-Option Data contains an "Interface Attributes (Type 2)" option encoded as follows:

- omIndex is set to an 8-bit integer value corresponding to a specific underlying interface the same as specified above for the OMNI option S/T-omIndex field. The OMNI options of a same message may include multiple Interface Attributes Sub-Options, with each distinct omIndex value pertaining to a different underlying interface. The OMNI option will often include an Interface Attributes Sub-Option with the same omIndex value that appears in the S/T-omIndex. 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.

- omType is set to an 8-bit integer value corresponding to the underlying interface identified by omIndex. 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 omIndex.

- 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...
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 15; 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 1, L2ADDR is the INET encapsulation address for the Source/Target Client itself; otherwise L2ADDR is the address of the Proxy/Server named in the LHS.

  - When the next most significant bit (i.e., "Mode") is set to 1, the Framework node is (likely) located behind an INET Network Address Translator (NAT); otherwise, it is on the open INET.

  - When the least significant bit (i.e., "Type") is set to 0, L2ADDR includes a UDP Port Number followed by an IPv4 address; otherwise, it includes a UDP Port Number followed by an IPv6 address.

+ LHS - the 32 bit MSID of the Last Hop Proxy/Server 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 2 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 network byte order, and in ones-complement "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 16. 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 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* 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 15). 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.

12.1.5. Traffic Selector

<table>
<thead>
<tr>
<th>S-Type=4</th>
<th>Sub-length=N</th>
<th>omIndex</th>
</tr>
</thead>
</table>

Figure 18: Traffic Selector

- Sub-Type is set to 4. If multiple instances appear in OMNI options of the same message all are processed, i.e., even if the same omIndex value appears multiple times.

- Sub-Length is set to N (from 1 to 2034) that encodes the number of Sub-Option Data octets that follow.

- Sub-Option Data contains a 1 octet omIndex encoded exactly as specified in Section 12.1.3, followed by an N-1 octet traffic selector formatted per [RFC6088] beginning with the "TS Format" field. The largest traffic selector for a given omIndex is therefore 2033 octets.

12.1.6. MS-Register
12.1.6. MS-Register

- Sub-Type is set to 5. If multiple instances appear in OMNI options of the same message 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.

- Sub-Length is set to 4n, with 508 as the maximum value for n. The length of the Sub-Option Data section is therefore limited to 2032 octets.

- 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.

12.1.7. MS-Release

- Sub-Type is set to 6. If multiple instances appear in OMNI options of the same message 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.

- Sub-Length is set to 4n, with 508 as the maximum value for n. The length of the Sub-Option Data section is therefore limited to 2032 octets.

- A list of n 4 octet MSIDs is included in the following 4n octets.
Sub-Type is set to 6. If multiple instances appear in OMNI options of the same message 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.

Sub-Length is set to 4n, with 508 as the maximum value for n. The length of the Sub-Option Data section is therefore limited to 2032 octets.

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.

12.1.8. Geo Coordinates

Sub-Type is set to 7. If multiple instances appear in OMNI options of the same message the first is processed and all others are ignored.

Sub-Length is set to N (from 0 to 2034) that encodes the number of Sub-Option Data octets that follow.

A set of Geo Coordinates of maximum length 2034 octets. Format(s) to be specified in future documents; should include Latitude/Longitude, plus any additional attributes such as altitude, heading, speed, etc.


The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) sub-option may be included in the OMNI options of RS messages sent by MNs and RA messages returned by MSEs. ARs that act as proxys to forward RS/RA messages between MNs and MSEs also forward DHCPv6 sub-options unchanged and do not process DHCPv6 sub-options themselves. Note that DHCPv6 message sub-option integrity is protected by the Checksum included in the IPv6 ND message header.
Figure 22: DHCPv6 Message Sub-option

- Sub-Type is set to 8. If multiple instances appear in OMNI options of the same message the first is processed and all others are ignored.

- Sub-Length is set to N (from 4 to 2034) that encodes the number of Sub-Option Data octets that follow. The 'msg-type' and 'transaction-id' fields are always present; hence, the length of the DHCPv6 options is restricted to 2030 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.

12.1.10. Host Identity Protocol (HIP) Message

The Host Identity Protocol (HIP) Message sub-option may be included in the OMNI options of RS messages sent by MNs and RA messages returned by ARs. ARs that act as proxys authenticate and remove HIP messages in RS messages they forward from a MN to an MSE. ARs that act as proxys insert and sign HIP messages in the RA messages they forward from an MSE to a MN.

The HIP message sub-option may also be included in any IPv6 ND message that may traverse an open Internetwork, i.e., where link-layer authentication is not already assured by lower layers.
Figure 23: HIP Message Sub-option

- Sub-Type is set to 9. If multiple instances appear in OMNI options of the same message the first is processed and all others are ignored.

- Sub-Length is set to N, i.e., the length of the option in octets beginning immediately following the Sub-Length field and extending to the end of the HIP parameters. The length of the entire HIP message is therefore restricted to 2034 octets.

- The HIP message is coded exactly as specified in Section 5 of [RFC7401], except that the OMNI "Sub-Type" and "Sub-Length" fields replace the first 2 octets of the HIP message header (i.e., the Next Header and Header Length fields). Note that, since the IPv6 ND message header already includes a Checksum, the HIP message Checksum field is set to 0 on transmission and ignored on reception. (The Checksum field is still included to retain the [RFC7401] message format.)

12.1.11. Reassembly Limit

The Reassembly Limit sub-option may be included in the OMNI options of IPv6 ND messages. The message consists of a 14-bit Reassembly Limit value, followed by two flag bits (H, L) optionally followed by...
an (N-2)-octet leading portion of an OAL First Fragment that triggered the message.

Figure 24: Reassembly Limit

- Sub-Type is set to 10. If multiple instances appear in OMNI options of the same message the first occurring "hard" and "soft" Reassembly Limit values are accepted, and any additional Reassembly Limit values are ignored.

- Sub-Length is set to 2 if no OAL First Fragment is included, or to a value N greater than 2 if an OAL First Fragment is included.

- A 14-bit Reassembly Limit follows, and includes a value between 1500 and 9180. If any other value is included, the sub-option is ignored. The value indicates the hard or soft limit for original IP packets that the source of the message is currently willing to reassemble; the source may increase or decrease the hard or soft limit at any time through the transmission of new IPv6 ND messages. Until the first IPv6 ND message with a Reassembly Limit sub-option arrives, OMNI nodes assume initial default hard/soft limits of 9180 bytes (i.e., the OMNI interface MRU). After IPv6 ND messages with Reassembly Limit sub-options arrive, the OMNI node retains the most recent hard/soft limit values until new IPv6 ND messages with different values arrive.

- The 'H' flag is set to 1 if the Reassembly Limit is a "Hard" limit, and set to 0 if the Reassembly Limit is a "Soft" limit.

- The 'L' flag is set to 1 if an OAL First Fragment corresponding to a reassembly loss event was included; otherwise set to 0.

- If N is greater than 2, the remainder of the Reassembly Limit sub-option encodes the leading portion of an OAL First Fragment that prompted this IPv6 ND message. The first fragment is included beginning with the OAL IPv6 header, and continuing with as much of the fragment payload as possible without causing the IPv6 ND message to exceed the minimum IPv6 MTU. (Note that only the OAL...
First Fragment is consulted regardless of its size, and without waiting for additional fragments.)

12.1.12. Fragmentation Report

The Fragmentation Report may be included in the OMNI options of uNA messages sent from an OAL destination to an OAL source. The message consists of \((N / 8)\)-many (Identification, Bitmap)-tuples which include the Identification values of OAL fragments received plus a Bitmap marking the ordinal positions of individual fragments received and fragments missing.

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S-Type=11|   Sub-Length = N    | Identification #1 (bits 0 -15) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Identification #1 (bits 15-31) |    Bitmap #1 (bits 0 - 15)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Bitmap #1 (bits 16-31)  | Identification #2 (bits 0 -15) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Identification #2 (bits 15-31) |    Bitmap #2 (bits 0 - 15)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Bitmap #2 (bits 16-31)  | Identification #3 (bits 0 -15) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Identification #3 (bits 15-31) |    Bitmap #3 (bits 0 - 15)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Bitmap #3 (bits 16-31)  |             ...               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+             ...               +
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 25: Fragmentation Report

- Sub-Type is set to 11. If multiple instances appear in OMNI options of the same message all are processed.
- Sub-Length is set to \(N\), i.e., the length of the option in octets beginning immediately following the Sub-Length field and extending to the end of the ICMPv6 error message body. \(N\) must be an integral multiple of 8 octets; otherwise, the sub-option is ignored. The length of the entire sub-option should not cause the entire IPv6 ND message to exceed the minimum MPS.
- Identification (i) includes the IPv6 Identification value found in the Fragment Header of a received OAL fragment. (Only those Identification values included represent fragments for which loss was unambiguously observed; any Identification values not included
correspond to fragments that were either received in their
entirety or are still in transit.)

- Bitmap (i) includes an ordinal checklist of fragments, with each
  bit set to 1 for a fragment received or 0 for a fragment missing.
  For example, for a 20-fragment fragmented OAL packet with ordinal
  fragments #3, #10, #13 and #17 missing and all other fragments
  received, the bitmap would encode:

  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
  +--------------------------------------------------+
  |1|1|0|1|1|1|0|1|1|0|1|1|0|1|1|0|1|1|0|1|1|0|0|0|...
  +--------------------------------------------------+

  Figure 26

  (Note that loss of an OAL atomic fragment is indicated by a
  Bitmap(i) with all bits set to 0.)

12.1.13. Node Identification

- Sub-Type is set to 12. If multiple instances appear in OMNI
  options of the same IPv6 ND message the first instance of a
  specific ID-Type is processed and all other instances of the same
  ID-Type are ignored. (Note therefore that it is possible for a
  single IPv6 ND message to convey multiple Node Identifications -
  each having a different ID-Type.)

- Sub-Length is set to N (from 1 to 2034) that encodes the number of
  Sub-Option Data octets that follow. The ID-Type field is always
  present; hence, the maximum Node Identification Value length is
  2033 octets.

- ID-Type is a 1 octet field that encodes the type of the Node
  Identification Value. The following ID-Type values are currently
  defined:
0 - Universally Unique IDentifier (UUID) [RFC4122]. Indicates that Node Identification Value contains a 16 octet UUID.

1 - Host Identity Tag (HIT) [RFC7401]. Indicates that Node Identification Value contains a 16 octet HIT.

2 - Hierarchical HIT (HHIT) [I-D.ietf-drip-rid]. Indicates that Node Identification Value contains a 16 octet HHIT.

3 - Network Access Identifier (NAI) [RFC7542]. Indicates that Node Identification Value contains an N-1 octet NAI.

4 - Fully-Qualified Domain Name (FQDN) [RFC1035]. Indicates that Node Identification Value contains an N-1 octet FQDN.

5 - 252 - Unassigned.

253-254 - Reserved for experimentation, as recommended in [RFC3692].

255 - reserved by IANA.

Node Identification Value is an (N - 1) octet field encoded according to the appropriate the "ID-Type" reference above.

When a Node Identification Value is needed for DHCPv6 messaging purposes, it is encoded as a DHCP Unique IDentifier (DUID) using the "DUID-EN for OMNI" format with enterprise number 45282 (see: Section 25) as shown in Figure 28:

```
+----------------+-----------------+-----------------+
|     EN (high bits == 0)     |     ID-Type    |               |
|     EN (low bits = 45282)   |                |               |
.                    Node Identification Value .
+----------------+-----------------+-----------------+
```

Figure 28: DUID-EN for OMNI Format

In this format, the ID-Type and Node Identification Value fields are coded exactly as in Figure 27 following the 6 octet DUID-EN header, and the entire "DUID-EN for OMNI" is included in a DHCPv6 message per [RFC8415].
12.1.14. Sub-Type Extension

Since the Sub-Type field is only 5 bits in length, future specifications of major protocol functions may exhaust the remaining Sub-Type values available for assignment. This document therefore defines Sub-Type 30 as an "extension", meaning that the actual sub-option type is determined by examining a 1 octet "Extension-Type" field immediately following the Sub-Length field. The Sub-Type Extension is formatted as shown in Figure 29:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S-Type=30|     Sub-length=N    | Extension-Type|               ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           ˜                       Extension-Type Body                     ˜
              ˜                   Extension-Type|               ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 29: Sub-Type Extension

- **Sub-Type** is set to 30. If multiple instances appear in OMNI options of the same message all are processed, where each individual extension defines its own policy for processing multiple of that type.

- **Sub-Length** is set to N (from 1 to 2034) that encodes the number of Sub-Option Data octets that follow. The Extension-Type field is always present; hence, the maximum Extension-Type Body length is 2033 octets.

- **Extension-Type** contains a 1 octet Sub-Type Extension value between 0 and 255.

- **Extension-Type Body** contains an N-1 octet block with format defined by the given extension specification.

Extension-Type values 2 through 252 are available for assignment by future specifications, which must also define the format of the Extension-Type Body and its processing rules. Extension-Type values 253 and 254 are reserved for experimentation, as recommended in [RFC3692], and value 255 is reserved by IANA. Extension-Type values 0 and 1 are defined in the following subsections:
12.1.14.1. RFC4380 UDP/IP Header Option

- Sub-Type is set to 30.
- Sub-Length is set to N (from 2 to 2034) that encodes the number of Sub-Option Data octets that follow. The Extension-Type and Header Type fields are always present; hence, the maximum-length Header Option Value is 2032 octets.
- Extension-Type is set to 0. Each instance encodes exactly one header option per Section 5.1.1 of [RFC4380], with the leading '0' octet omitted and the following octet coded as Header Type. If multiple instances of the same Header Type appear in OMNI options of the same message the first instance is processed and all others are ignored.
- Header Type and Header Option Value are coded exactly as specified in Section 5.1.1 of [RFC4380]; the following types are currently defined:
  * 0 - Origin Indication (IPv4) - value coded per Section 5.1.1 of [RFC4380].
  * 1 - Authentication Encapsulation - value coded per Section 5.1.1 of [RFC4380].
  * 2 - Origin Indication (IPv6) - value coded per Section 5.1.1 of [RFC4380], except that the address is a 16-octet IPv6 address instead of a 4-octet IPv4 address.
- Header Type values 3 through 252 are available for assignment by future specifications, which must also define the format of the Header Option Value and its processing rules. Header Type values 253 and 254 are reserved for experimentation, as recommended in [RFC3692], and value 255 is Reserved by IANA.
12.1.14.2. RFC6081 UDP/IP Trailer Option

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+-----------------------------------------------+-----------------------------------------------+-----------------------------------------------+-----------------------------------------------+
| S-Type=30 | Sub-length=N | Ext-Type=1 | Trailer Type |
|-----------------------------------------------+-----------------------------------------------+-----------------------------------------------+-----------------------------------------------+
|                                                | Trailer Option Value                          |                                               |
|                                                | +-----------------------------------------------+
|                                                | +-----------------------------------------------+

Figure 31: RFC6081 UDP/IP Trailer Option (Extension-Type 1)

- Sub-Type is set to 30.
- Sub-Length is set to N (from 2 to 2034) that encodes the number of Sub-Option Data octets that follow. The Extension-Type and Trailer Type fields are always present; hence, the maximum-length Trailer Option Value is 2032 octets.
- Extension-Type is set to 1. Each instance encodes exactly one trailer option per Section 4 of [RFC6081]. If multiple instances of the same trailer type appear in OMNI options of the same message the first instance is processed and all others ignored.
- Trailer Type and Trailer Option Value are coded exactly as specified in Section 4 of [RFC6081]; the following Trailer Types are currently defined:
  * 0 - Unassigned
  * 1 - Nonce Trailer - value coded per Section 4.2 of [RFC6081].
  * 2 - Unassigned
  * 3 - Alternate Address Trailer (IPv4) - value coded per Section 4.3 of [RFC6081].
  * 4 - Neighbor Discovery Option Trailer - value coded per Section 4.4 of [RFC6081].
  * 5 - Random Port Trailer - value coded per Section 4.5 of [RFC6081].
  * 6 - Alternate Address Trailer (IPv6) - value coded per Section 4.3 of [RFC6081], except that each address is a 16-octet IPv6 address instead of a 4-octet IPv4 address.
o Trailer Type values 7 through 252 are available for assignment by future specifications, which must also define the format of the Trailer Option Value and its processing rules. Trailer Type values 253 and 254 are reserved for experimentation, as recommended in [RFC3692], and value 255 is Reserved by IANA.

13. 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].

14. Multilink Conceptual Sending Algorithm

The MN’s IPv6 layer selects the outbound OMNI interface according to SBM considerations when forwarding original IP 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 15, and may configure more-specific routes discovered through means outside the scope of this specification.

After an original IP 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 15). 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 determines the multilink forwarding profile for selecting the neighbor’s underlying interfaces.

When the OMNI interface sends an original IP packet over a selected outbound underlying interface, the OAL employs encapsulation and fragmentation as discussed in Section 5, then performs *NET encapsulation 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 15.1. (Note that the OAL MAY employ packing when multiple original IP packets and/or control messages are available for forwarding to the same OAL destination.)

OMNI interface multilink service designers MUST observe the BCP guidance in Section 15 [RFC3819] in terms of implications for reordering when original IP packets from the same flow may be spread across multiple underlying interfaces having diverse properties.

14.1. Multiple OMNI Interfaces

MNIs 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 an original IP packet’s 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 packet 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 packet, 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).

14.2. MN<-->AR Traffic Loop Prevention

After an AR has registered an MNP for a MN (see: Section 15), 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:

- 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.

15. 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 20).

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’s OMNI interface sends initial RS messages
over an UP underlying interface with its MNP-LLA as the source and with destination set to link-scoped All-Routers multicast (ff02::2) [RFC4291]. The OMNI interface includes an OMNI option per Section 12 with a Preflen assertion, Interface Attributes appropriate for underlying interfaces, MS-Register/Release sub-options containing MSID values, Reassembly Limits, an authentication sub-option and with any other necessary OMNI sub-options (e.g., a Node Identification sub-option as an identity for the MN). The OMNI interface then sets the S/T-omIndex field to the index of the underlying interface over which the RS message is sent.

The OMNI interface then sends the RS over the underlying interface using OAL encapsulation and fragmentation if necessary. If OAL encapsulation is used for RS messages sent over an INET interface, the entire RS message must appear within a single carrier packet so that it can be authenticated without requiring reassembly. The OMNI interface selects an unpredictable initial Identification value per Section 6.5, sets the OAL source address to the ULA corresponding to the RS source and sets the OAL destination to site-scoped All-Routers multicast (ff05::2) then sends the message.

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 12.

The AR’s OMNI interface 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-omIndex 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 Node Identification 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:

- PIOs with (A; L=0) that include MSPs for the link [RFC8028].
- RIOs [RFC4191] with more-specific routes.
an MTU option that specifies the maximum acceptable packet size for this underlying interface.

The OMNI interface then sends the RA, using OAL encapsulation/fragmentation with the same Identification value that appeared in the RS message OAL header. The OMNI interface sets the OAL source address to the ULA corresponding to the RA source and sets the OAL destination to the ULA corresponding to the RA destination. The AR MAY also send periodic and/or event-driven unsolicited RA messages per [RFC4861]. In that case, the S/T-omIndex field in the OMNI option 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 omIndex 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.

15.1. 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 MANET/VANET 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 its MNP-LLA (or to the unspecified address (::) if it does not yet have an MNP-LLA), and with destination set to link-scoped All-Routers multicast the same as discussed above. The OMNI interface then employs OAL encapsulation and fragmentation, and sets the OAL source address to the ULA corresponding to the RS source (or to a Temporary ULA if the RS source was the unspecified address (::)) and sets the OAL destination to site-scoped All-Routers multicast (ff05::2). For IPv6-enabled *NETs, the MN then encapsulates the message in UDP/IPv6 headers with source address set to the underlying interface address (or to the ULA that would be used for OAL encapsulation if the underlying interface
does not yet have an address) and sets the destination to either a unicast or anycast address of an AR. For IPv4-only *NETs, the MN instead encapsulates the RS message in UDP/IPv4 headers with source address set to the IPv4 address of the underlying interface 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 (while performing OAL reassembly, if necessary) and coordinates with the MS the same as for an ordinary link-local RS, since the network layer 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. The AR then performs OAL encapsulation and fragmentation, with OAL source set to its own ADM-ULA and destination set to the ULA corresponding to the RA source. The AR then encapsulates the message in UDP/IPv4 or UDP/IPv6 headers with source address set to its own 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 for the MN and AR to statelessly translate the IPv6 LLAs into ULAs and forward the RS/RA 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 mandates 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].

15.2. MS-Register and MS-Release List Processing

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.

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 "anycast" 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:

- 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).
- Next, for each MSID in the MS-Register list, remove all matching MSIDs from the MS-Release list.
- Next, proceed as follows:
  - If the AR’s own MSID or the value 0 appears in the MS-Register list, 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 an unsolicited NA (uNA) message to each MSID in the MS-Release list with the MS-Register/Release lists omitted and with an OMNI option with S/T-omIndex set to 0.
  - Otherwise, 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 OAL 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 with an OMNI option with S/T-omIndex set to 0. The Register MSE then sends an RA message back to the (Proxy) AR wrapped in an OAL 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 sent by the first-hop AR or by a Register MSE) will include an OMNI option and an OAL encapsulation header with the ADM-ULA of the Register MSE as the source and the ADM-ULA of the Release MSE 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.

15.3. DHCPv6-based Prefix Registration

When a MN is not pre-provisioned with an MNP-LLA (or, when the MN requires additional MNP delegations), it requests the MSE 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 MSE select MNPs, it sends an RS message with source set to the unspecified address (::) if it has no MNP_LLAs. If the MN requires only a single MNP delegation, it can then include a Node Identification sub-option in the OMNI option and set Preflen to the length of the desired MNP. If the MN requires multiple MNP delegations and/or more complex DHCPv6 services, it instead includes a DHCPv6 Message sub-option containing a Client Identifier, one or more IA_PD options and a Rapid Commit option then sets the 'msg-type' field to "Solicit", and includes a 3 octet 'transaction-id’. The MN then sets the RS destination to All-Routers multicast and sends the message using OAL encapsulation and fragmentation if necessary as discussed above.

When the MSE receives the RS message, it performs OAL reassembly if necessary. Next, if the RS source is the unspecified address (::) and/or the OMNI option includes a DHCPv6 message sub-option, the MSE acts as a "Proxy DHCPv6 Client" in a message exchange with the locally-resident DHCPv6 server. If the RS did not contain a DHCPv6 message sub-option, the MSE generates a DHCPv6 Solicit message on behalf of the MN using an IA_PD option with the prefix length set to the OMNI header Preflen value and with a Client Identifier formed from the OMNI option Node Identification sub-option; otherwise, the MSE uses the DHCPv6 Solicit message contained in the OMNI option. The MSE then sends the DHCPv6 message to the DHCPv6 Server, which delegates MNPs and returns a DHCPv6 Reply message with PD parameters. (If the MSE 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. In the process, the MSE packs any state information needed to return an RA to the MN in the Relay-forward Interface ID option so that the information will be echoed back in the Relay-reply.)

When the MSE receives the DHCPv6 Reply, it adds routes to the routing system and creates MNP-LLAs based on the delegated MNPs. The MSE then sends an RA back to the MN with the DHCPv6 Reply message.
included in an OMNI DHCPv6 message sub-option if and only if the RS message had included an explicit DHCPv6 Solicit. If the RS message source was the unspecified address (::), the MSE includes one of the (newly-created) MNP-LLAs as the RA destination address and sets the OMNI option Preflen accordingly; otherwise, the MSE includes the RS source address as the RA destination address. The MSE then sets the RA source address to its own ADM-LLA then performs OAL encapsulation and fragmentation and sends the RA to the MN. When the MN receives the RA, it reassembles and discards the OAL encapsulation, 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: After a MN performs a DHCPv6-based prefix registration exchange with a first MSE, it would need to repeat the exchange with each additional MSE it registers with. In that case, the MN supplies the MNP delegation information received from the first MSE when it engages the additional MSEs.

16. 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.

17. 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.

18. 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.

19. 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.

20. OMNI Interfaces on Open Internetworks

OMNI interfaces configured over IPv6-enabled underlying interfaces on an open Internetwork 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 (although they may receive IPv4 RA messages [RFC1256]). 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 an open Internetwork 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 per [RFC6081][RFC4380] and HIP-based message authentication per [RFC7401].

OMNI interfaces use UDP service port number 8060 (see: Section 25.10 and Section 3.6 of [I-D.templin-intarea-6706bis]) according to the simple UDP/IP encapsulation format specified in [RFC4380] for both IPv4 and IPv6 underlying interfaces. OMNI interfaces do not include the UDP/IP header/trailer extensions specified in [RFC4380][RFC6081], but may include them as OMNI sub-options instead when necessary. Since the OAL includes an integrity check over the OAL packet, OAL sources selectively disable UDP checksums for OAL packets that do not require UDP/IP address integrity, but enable UDP checksums for others including non-OAL packets, IPv6 ND messages used to establish link-layer addresses, etc. If the OAL source discovers that packets with
UDP checksums disabled are being dropped in the path it should enable UDP checksums in future packets. Further considerations for UDP encapsulation checksums are found in [RFC6935][RFC6936].

For "Vehicle-to-Infrastructure (V2I)" coordination, the MN codes an authentication sub-option in an OMNI option of an IPv6 RS message and the AR responds with an authentication sub-option in an OMNI option of an IPv6 RA message. HIP security services can be applied per [RFC7401] using the RS/RA messages as simple "shipping containers" to convey the HIP parameters. Alternatively, a simple Hashed Message Authentication Code (HMAC) can be included in the manner specified in [RFC4380]. For "Vehicle-to-Vehicle (V2V)" coordination, two MNs can coordinate directly with one another with HIP "Initiator/Responder" messages coded in OMNI options of IPv6 NS/NA messages. In that case, a four-message HIP exchange (i.e., two back-to-back NS/NA exchanges) may be necessary for the two MNs to attain mutual authentication.

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 15 and NS/NA messages used for route optimization and mobility (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 traffic with basic source address-based data origin verification. Data plane communications via OMNI interfaces that connect over open Internetworks without an explicit VPN should therefore employ transport- or higher-layer security to ensure integrity and/or confidentiality.

OMNI interfaces configured over open Internetworks are often located behind NATs. The OMNI interface accommodates NAT traversal using UDP/IP encapsulation and the mechanisms discussed in [I-D.templin-intarea-6706bis]. To support NAT determination, ARs include an Origin Indication sub-option in RA messages sent in response to RS messages received from a Client via UDP/IP encapsulation.

Note: Following the initial HIP Initiator/Responder exchange, OMNI interfaces configured over open Internetworks maintain HIP associations through the transmission of IPv6 ND messages that include OMNI options with HIP "Update" and "Notify" messages. OMNI interfaces use the HIP "Update" message when an acknowledgement is required, and use the "Notify" message in unacknowledged isolated IPv6 ND messages (e.g., unsolicited NAs). When HMAC authentication is used instead of HIP, the MN and AR exchange all IPv6 ND messages with HMAC signatures included based on a shared-secret.
Note: ARs that act as proxys on an open Internetwork authenticate and remove authentication OMNI sub-options from IPv6 ND messages they forward from a MN, and insert and sign authentication Origin Indication sub-options in IPv6 ND messages they forward from the network to the MN. Conversely, ARs that act as proxys forward without processing any DHCPv6 information in RS/RA message exchanges between MNs and MSEs. The AR is therefore responsible for MN authentication while the MSE is responsible for registering/delegating MNPs.

Note: A simpler arrangement is possible when the AR also acts as a MSE itself, i.e., when the proxy and MSE functions are combined on a single physical or logical platform.

21. 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 15.3 allows OMNI MNs that desire time-varying MNPs to obtain short-lived prefixes to send RS messages with source set to the unspecified address (::) and/or 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.

22. (H)HITs and Temporary ULAs

MNIs that generate (H)HITs but do not have pre-assigned MNPs can request MNP delegations by issuing IPv6 ND messages that use the (H)HIT instead of a Temporary ULA. In particular, when a MN creates an RS message it can set the source to the unspecified address (::) and destination to All-Routers multicast. The IPv6 ND message includes an OMNI option with a HIP "Initiator" message sub-option, and need not include a Node Identification sub-option since the MN's HIT appears in the HIP message. The MN then encapsulates the message in an IPv6 header with the (H)HIT as the source address and with destination set to either a unicast or anycast ADM-ULA. The MN then sends the message to the AR as specified in Section 15.1.
When the AR receives the message, it notes that the RS source was the unspecified address (::), then examines the RS encapsulation source address to determine that the source is a (H)HIT and not a Temporary ULA. The AR next invokes the DHCPv6 protocol to request an MNP prefix delegation while using the HIT as the Client Identifier, then prepares an RA message with source address set to its own ADM-LLA and destination set to the MNP-LLA corresponding to the delegated MNP. The AR next includes an OMNI option with a HIP "Responder" message and any DHCPv6 prefix delegation parameters. The AR then finally encapsulates the RA in an IPv6 header with source address set to its own ADM-ULA and destination set to the (H)HIT from the RS encapsulation source address, then returns the encapsulated RA to the MN.

MNs can also use (H)HITs and/or Temporary ULAs for direct MN-to-MN communications outside the context of any OMNI link supporting infrastructure. When two MNs encounter one another they can use their (H)HITs and/or Temporary ULAs as original IPv6 packet source and destination addresses to support direct communications. MNs can also inject their (H)HITs and/or Temporary ULAs into a MANET/VANET routing protocol to enable multihop communications. MNs can further exchange IPv6 ND messages (such as NS/NA) using their (H)HITs and/or Temporary ULAs as source and destination addresses. Note that the HIP security protocols for establishing secure neighbor relationships are based on (H)HITs. Temporary ULAs instead use the HMAC authentication service specified in [RFC4380].

Lastly, when MNs are within the coverage range of OMNI link infrastructure a case could be made for injecting (H)HITs and/or Temporary ULAs into the global MS routing system. For example, when the MN sends an RS to a MSE it could include a request to inject the (H)HIT / Temporary ULA into the routing system instead of requesting an MNP prefix delegation. This would potentially enable OMNI link-wide communications using only (H)HITs or Temporary ULAs, and not MNPs. This document notes the opportunity, but makes no recommendation.

23. Address Selection

OMNI MNs use LLAs only for link-scoped communications on the OMNI link. Typically, MNs use LLAs as source/destination IPv6 addresses of IPv6 ND messages, but may also use them for addressing ordinary original IP packets exchanged with an OMNI link neighbor.

OMNI MNs use MNP-ULAs as source/destination IPv6 addresses in the encapsulation headers of OAL packets. OMNI MNs use Temporary ULAs for OAL addressing when an MNP-ULA is not available, or as source/destination IPv6 addresses for communications within a MANET/VANET
OMNI MNs use HITs instead of Temporary ULAs when operation outside the context of a specific ULA domain and/or source address attestation is necessary.

OMNI MNs use MNP-based GUAs as original IP packet source and destination addresses for communications with Internet destinations when they are within range of OMNI link supporting infrastructure that can inject the MNP into the routing system.

24. Error Messages

An OAL destination or intermediate node may need to return ICMPv6 error messages (e.g., Destination Unreachable, Packet Too Big, Time Exceeded, etc.) [RFC4443] to an OAL source. Since ICMPv6 error messages do not themselves include authentication codes, the OAL includes the ICMPv6 error message as an OMNI sub-option in an IPv6 ND uNA message. The OAL also includes a HIP message sub-option if the uNA needs to travel over an open Internetwork.

25. IANA Considerations

The following IANA actions are requested:

25.1. "IEEE 802 Numbers" Registry

The IANA is instructed to allocate an official Ether Type number TBD1 from the 'ieee-802-numbers' registry for User Datagram Protocol (UDP) encapsulation on Ethernet networks. Guidance is found in [RFC7042].

25.2. "IPv6 Neighbor Discovery Option Formats" Registry

The IANA is instructed to allocate an official Type number TBD2 from the "IPv6 Neighbor Discovery Option Formats" registry for the OMNI option. Implementations set Type to 253 as an interim value [RFC4727].

25.3. "Ethernet Numbers" Registry

The IANA is instructed to allocate one Ethernet unicast address TBD3 (suggested value '00-52-14') in the 'ethernet-numbers' registry under "IANA Unicast 48-bit MAC Addresses" as follows:

<table>
<thead>
<tr>
<th>Addresses</th>
<th>Usage</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-52-14</td>
<td>Overlay Multilink Network (OMNI) Interface</td>
<td>[RFCXXXX]</td>
</tr>
</tbody>
</table>

Figure 32: IANA Unicast 48-bit MAC Addresses
25.4. "ICMPv6 Code Fields: Type 2 - Packet Too Big" Registry

The IANA is instructed to assign two new Code values in the "ICMPv6 Code Fields: Type 2 - Packet Too Big" registry. The registry should appear as follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PTB Hard Error</td>
<td>[RFC4443]</td>
</tr>
<tr>
<td>1</td>
<td>PTB Soft Error (loss)</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>2</td>
<td>PTB Soft Error (no loss)</td>
<td>[RFCXXXX]</td>
</tr>
</tbody>
</table>

Figure 33: ICMPv6 Code Fields: Type 2 - Packet Too Big Values

(Note: this registry also to be used to define values for setting the "unused" field of ICMPv4 "Destination Unreachable - Fragmentation Needed" messages.)

25.5. "OMNI Option Sub-Type Values" (New Registry)

The OMNI option defines a 5-bit Sub-Type field, for which IANA is instructed to create and maintain a new registry entitled "OMNI Option Sub-Type Values". Initial values are given below (future assignments are to be made through Standards Action [RFC8126]):

<table>
<thead>
<tr>
<th>Value</th>
<th>Sub-Type name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pad1</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>1</td>
<td>PadN</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>2</td>
<td>Interface Attributes (Type 1)</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>3</td>
<td>Interface Attributes (Type 2)</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>4</td>
<td>Traffic Selector</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>5</td>
<td>MS-Register</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>6</td>
<td>MS-Release</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>7</td>
<td>Geo Coordinates</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>8</td>
<td>DHCPv6 Message</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>9</td>
<td>HIP Message</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>10</td>
<td>Reassembly Limit</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>11</td>
<td>Fragmentation Report</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>12</td>
<td>Node Identification</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>13-29</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Sub-Type Extension</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>31</td>
<td>Reserved by IANA</td>
<td>[RFCXXXX]</td>
</tr>
</tbody>
</table>

Figure 34: OMNI Option Sub-Type Values
25.6. "OMNI Node Identification ID-Type Values" (New Registry)

The OMNI Node Identification Sub-Option (see: Section 12.1.13) contains an 8-bit ID-Type field, for which IANA is instructed to create and maintain a new registry entitled "OMNI Node Identification ID-Type Values". Initial values are given below (future assignments are to be made through Expert Review [RFC8126]):

<table>
<thead>
<tr>
<th>Value</th>
<th>Sub-Type name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UUID</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>1</td>
<td>HIT</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>2</td>
<td>HHIT</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>3</td>
<td>Network Access Identifier</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>4</td>
<td>FQDN</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>5-252</td>
<td>Unassigned</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>253-254</td>
<td>Reserved for Experimentation</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>255</td>
<td>Reserved by IANA</td>
<td>[RFCXXXX]</td>
</tr>
</tbody>
</table>

Figure 35: OMNI Node Identification ID-Type Values

25.7. "OMNI Option Sub-Type Extension Values" (New Registry)

The OMNI option defines an 8-bit Extension-Type field for Sub-Type 30 (Sub-Type Extension), for which IANA is instructed to create and maintain a new registry entitled "OMNI Option Sub-Type Extension Values". Initial values are given below (future assignments are to be made through Expert Review [RFC8126]):

<table>
<thead>
<tr>
<th>Value</th>
<th>Sub-Type name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RFC4380 UDP/IP Header Option</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>1</td>
<td>RFC6081 UDP/IP Trailer Option</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>2-252</td>
<td>Unassigned</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>253-254</td>
<td>Reserved for Experimentation</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>255</td>
<td>Reserved by IANA</td>
<td>[RFCXXXX]</td>
</tr>
</tbody>
</table>

Figure 36: OMNI Option Sub-Type Extension Values

25.8. "OMNI RFC4380 UDP/IP Header Option" (New Registry)

The OMNI Sub-Type Extension "RFC4380 UDP/IP Header Option" defines an 8-bit Header Type field, for which IANA is instructed to create and maintain a new registry entitled "OMNI RFC4380 UDP/IP Header Option". Initial registry values are given below (future assignments are to be made through Expert Review [RFC8126]):
<table>
<thead>
<tr>
<th>Value</th>
<th>Sub-Type name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Origin Indication (IPv4)</td>
<td>[RFC4380]</td>
</tr>
<tr>
<td>1</td>
<td>Authentication Encapsulation</td>
<td>[RFC4380]</td>
</tr>
<tr>
<td>2</td>
<td>Origin Indication (IPv6)</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>3-252</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>253-254</td>
<td>Reserved for Experimentation</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>255</td>
<td>Reserved by IANA</td>
<td>[RFCXXXX]</td>
</tr>
</tbody>
</table>

Figure 37: OMNI RFC4380 UDP/IP Header Option

25.9. "OMNI RFC6081 UDP/IP Trailer Option" (New Registry)

The OMNI Sub-Type Extension for "RFC6081 UDP/IP Trailer Option" defines an 8-bit Trailer Type field, for which IANA is instructed to create and maintain a new registry entitled "OMNI RFC6081 UDP/IP Trailer Option". Initial registry values are given below (future assignments are to be made through Expert Review [RFC8126]):

<table>
<thead>
<tr>
<th>Value</th>
<th>Sub-Type name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Nonce</td>
<td>[RFC6081]</td>
</tr>
<tr>
<td>2</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Alternate Address (IPv4)</td>
<td>[RFC6081]</td>
</tr>
<tr>
<td>4</td>
<td>Neighbor Discovery Option</td>
<td>[RFC6081]</td>
</tr>
<tr>
<td>5</td>
<td>Random Port</td>
<td>[RFC6081]</td>
</tr>
<tr>
<td>6</td>
<td>Alternate Address (IPv6)</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>7-252</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>253-254</td>
<td>Reserved for Experimentation</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>255</td>
<td>Reserved by IANA</td>
<td>[RFCXXXX]</td>
</tr>
</tbody>
</table>

Figure 38: OMNI RFC6081 Trailer Option

25.10. Additional Considerations

The IANA has assigned the UDP port number "8060" for an earlier experimental version of AERO [RFC6706]. This document together with [I-Dtemplin-intarea-6706bis] reclaims the UDP port number "8060" for 'aero' as the service port for UDP/IP encapsulation. (Note that, although [RFC6706] was not widely implemented or deployed, any messages coded to that specification can be easily distinguished and ignored since they use an invalid ICMPv6 message type number '0'.) The IANA is therefore instructed to update the reference for UDP port number "8060" from "RFC6706" to "RFCXXXX" (i.e., this document).

The IANA has assigned a 4 octet Private Enterprise Number (PEN) code "45282" in the "enterprise-numbers" registry. This document is the
normative reference for using this code in DHCP Unique IDentifiers based on Enterprise Numbers ("DUID-EN for OMNI Interfaces") (see: Section 11). The IANA is therefore instructed to change the enterprise designation for PEN code "45282" from "LinkUp Networks" to "Overlay Multilink Network Interface (OMNI)".

The IANA has assigned the ifType code "301 - omni - Overlay Multilink Network Interface (OMNI)" in accordance with Section 6 of [RFC8892]. The registration appears under the IANA "Structure of Management Information (SMI) Numbers (MIB Module Registrations) - Interface Types (ifType)" registry.

No further IANA actions are required.

26. 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. (Note however that when OAL encapsulation is used the (echoed) OAL Identification value can provide sufficient transaction confirmation.)

MN OMNI interfaces configured over secured ANET interfaces inherit the physical and/or link-layer security properties (i.e., "protected spectrum") of the connected ANETs. MN 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, the security services specified in [RFC7401] and/or [RFC4380] can be employed. 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.

Strong network layer security for control plane messages and forwarding path integrity for data plane messages between MSEs MUST be supported. In one example, the AERO service [I-D.templin-intarea-6706bis] constructs a spanning tree between MSEs 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.
Identity-based key verification infrastructure services such as iPSK may be necessary for verifying the identities claimed by MNs. This requirement should be harmonized with the manner in which (H)HITs are attested in a given operational environment.

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 6.9.

27. Implementation Status

AERO/OMNI Release-3.0.2 was tagged on October 15, 2020, and is undergoing internal testing. Additional internal releases expected within the coming months, with first public release expected end of 1H2021.

28. 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, Alok 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: Stuart Card, Michael Matyas, Robert Moskowitz, Madhu Niraula, Greg Saccone, Stephane Tamalet, Eric Vyncke. Pavel Drasil, Zdenek Jaron and Michal Skorepa are especially recognized for their many helpful ideas and suggestions. Madhuri Madhava Badgandi, Sean Dickson, Don Dillenburg, Joe Dudkowski, Vijayasarathy Rajagopalan, Ron Sackman and Katherine Tran are acknowledged for their hard work on the implementation and technical insights that led to improvements for 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. Many of the ideas in this document have further built on IETF experiences beginning as early as Y2K, with insights from colleagues including Brian Carpenter, Ralph Droms, Christian Huitema, Thomas Narten, Dave Thaler, Joe Touch, and many others who deserve recognition.

Early observations on IP fragmentation performance implications were noted in the 1986 Digital Equipment Corporation (DEC) "qe reset" investigation, where fragment bursts from NFS UDP traffic triggered hardware resets resulting in communication failures. Jeff Chase, Fred Glover and Chet Juzsczak of the Ultrix Engineering Group led the investigation, and determined that setting a smaller NFS mount block size reduced the amount of fragmentation and suppressed the resets. Early observations on L2 media MTU issues were noted in the 1988 DEC FDDI investigation, where Raj Jain, KK Ramakrishnan and Kathy Wilde represented architectural considerations for FDDI networking in general including FDDI/Ethernet bridging. Jeff Mogul (who led the IETF Path MTU Discovery working group) and other DEC colleagues who supported these early investigations are also acknowledged.

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.

This work is aligned with the Boeing Information Technology (BIT) Mobility Vision Lab (MVL) program.

29. References

29.1. Normative References


29.2. Informative References


[IPV6-GUA]


Templin & Whyman  Expires September 25, 2021
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Sub-Type=3 | Sub-length=N | omIndex | pmType | omIndex | pmType |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| 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 | P66 | P67 | P68 | ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 39: Example 1: Dense Simplex 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 TBD3 (see: Section 25). 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:

- Major clarifications on aspects such as "hard/soft" PTB error messages
- 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:

- MTU
- Support for multi-hop ANETS such as ISATAP.

Differences from draft-templin-6man-omni-interface-29 to draft-templin-6man-omni-interface-30:

- Moved link-layer addressing information into the OMNI option on a per-ifIndex basis
- Renamed "ifIndex-tuple" to "Interface Attributes"

Differences from draft-templin-6man-omni-interface-27 to draft-templin-6man-omni-interface-28:

- Updates based on implementation experience.

Differences from draft-templin-6man-omni-interface-25 to draft-templin-6man-omni-interface-26:

- Further clarification on "aggregate" RA messages.
- 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:
Safety-Based Multilink (SBM) and Performance-Based Multilink (PBM).

Differences from draft-templin-6man-omni-interface-18 to draft-templin-6man-omni-interface-19:
- SEND/CGA.

Differences from draft-templin-6man-omni-interface-17 to draft-templin-6man-omni-interface-18:
- Teredo

Differences from draft-templin-6man-omni-interface-14 to draft-templin-6man-omni-interface-15:
- Prefix length discussions removed.

Differences from draft-templin-6man-omni-interface-12 to draft-templin-6man-omni-interface-13:
- 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
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.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."
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 August 26, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction .................................................. 2
   1.1. BIER over L2/Tunnels .................................. 3
   1.2. Considerations of Requirements for BIER in IPv6 Networks 3
2. IPv6 Header .................................................... 5
   2.1. IPv6 Options Considerations ............................. 5
3. BIER Header .................................................... 6
4. IPv6 Encapsulation Advertisement .............................. 6
   4.1. Format .................................................... 6
   4.2. Inter-area prefix redistribution ........................ 7
5. IANA Considerations ............................................ 7
6. Security Considerations ........................................ 7
7. Acknowledgement ............................................... 7
8. References ...................................................... 7
   8.1. Normative References .................................... 8
   8.2. Informative References ................................... 8
Authors’ Addresses ................................................ 10

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:

- An operator mandates all traffic to be carried in IPv6.
- 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:

- **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.

- **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 based overlay services (e.g. MVPN/EVPN). One of the following methods can be used (relevant overlay signaling will be specified separately):

- **An ingress PE (which is a BFIR) can encapsulate customer packets with an IPv6 header (with optional fragmentation and ESP extension headers).** The destination address is a multicast locator plus the Func/Arg portion that identifies the service. 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.

- **Alternatively, since only the destination IPv6 address in the above-mentioned IPv6 header is used for service delimiting purpose, a new value can be assigned for the Proto field in the BIER header to indicate that an IPv6 address (instead of an entire IPv6 header) is added between the BIER header and original payload.**

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:

- BIER OAM specified in [I-D.ietf-bier-ping]
- BIER BFD specified in [I-D.ietf-bier-bfd]
- BIER Performance Measurement specified in [I-D.ietf-bier-pmmm-oam]
- BIER Path Maximum Transmission Unit Discovery specified in [I-D.ietf-bier-path-mtu-discovery]
- 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|   Type                  |   Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

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 "IPv6 address" type in the "BIER Next Protocol Identifiers" 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-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


8.2.  Informative References

[I-D.ietf-bier-bar-ipa]

[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-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-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-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-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., Bo, W., 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,

[I-D.zhang-bier-babel-extensions]

[I-D.zzhang-tsvwg-generic-transport-functions]


Authors’ Addresses
Zheng(Sandy) Zhang
ZTE Corporation
EMail: zhang.zheng@zte.com.cn

Zhaohui Zhang (editor)
Juniper Networks
EMail: zzhang@juniper.net

IJJsbrand Wijnands
Individual
EMail: ice@braindump.be