

Network Working Group
Internet Draft
Intended status: Standard
Expires: September 7, 2022

L. Dunbar
J. Kaippallimalil
Futurewei

March 7, 2022

IPv6 Solution for 5G Edge Computing Sticky Service
draft-dunbar-6man-5g-edge-compute-sticky-service-06

Abstract

This draft describes the IPv6-based solutions that can stick an application flow originated from a mobile device to the same ANYCAST server location when the mobile device moves from one 5G cell site to another.

Status of this Memo

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

This Internet-Draft will expire on April 7, 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction.....	3
1.1. 5G Edge Computing Background.....	3
1.2. 5G Edge Computing Network Properties.....	4
1.3. Problem #1: Discovery of Edge Application Server.....	5
1.4. Problem #2: sticking to original App Server.....	6
2. Conventions used in this document.....	7
3. Stick a Flow to an ANYCAST Server.....	9
4. Sticky flow for QUIC based Applications.....	9
5. Other Solutions within a Limited Domain.....	10
5.1. Use Case of 5G Edge Computing in a limited domain....	10
5.2. End Node Based Sticky Service Solution.....	10
5.2.1. Edge Controller Based Solution.....	11
5.3. Sticky Egress Address Discovery.....	12
5.4. Sticky-Dst-SubTLV in Destination Extension Header....	12
5.5. Processing at the Ingress router.....	13
6. Tunnel based Sticky Service Solution.....	13
6.1. Desired functions by the Network Controller.....	14
6.2. Ingress and Egress Routers Processing Behavior.....	14
6.3. A Solution without the Communication with 5G system..	16
6.4. A Solution that depends on the communication with 5G system.....	16
7. Expanding APN6 for Sticky Service information.....	17
7.1. Sticky Service ID encoded in the Application-aware ID	17

7.2. Sticky Service Sub-TLV encoded in APN6 Service-para option.....	18
8. Manageability Considerations.....	18
9. Security Considerations.....	18
10. IANA Considerations.....	18
11. References.....	18
11.1. Normative References.....	18
11.2. Informative References.....	19
12. Acknowledgments.....	20

1. Introduction

1.1. 5G Edge Computing Background

As described in [5G-EC-Metrics], one application in 5G Edge Computing environment can have multiple application servers hosted in different Edge Computing data centers close in proximity. Those Edge Computing (mini) data centers are usually very close to, or co-located with, 5G base stations, to minimize latency and optimize the performances.

When a mobile device sends packets using the destination address from a DNS reply or its own cache, the packets are carried by a GTP tunnel from the 5G eNB to the 5G UPF-PSA (User Plan Function - PDU Session Anchor). The UPF-PSA decapsulates the 5G GTP outer header and forwards the packets from the mobile devices to the Ingress router of the Edge Computing (EC) Local Data Network (LDN). The LDN for 5G EC, the IP Networks, is responsible for forwarding the packets to the intended destinations.

When the mobile device moves out of coverage of its current gNB (next-generation Node B) (gNB1), handover procedures are initiated, and the 5G SMF (Session Management Function) selects a new UPF-PSA. The standard handover procedures are described in 3GPP TS 23.501 and TS 23.502. When the handover process is complete, the mobile device might be anchored to a new UPF-PSA. 5G Session Management function (SMF) may maintain a path from the old UPF to the new UPF for a short period of time for SSC [Session and Service Continuity] mode 3 to make the handover process more seamless.

1.2. 5G Edge Computing Network Properties

In this document, 5G Edge Computing Network refers to multiple Local IP Data Networks (LDN) in one region that interconnect the Edge Computing mini-data centers. Those IP LDN networks are the N6 interfaces from 3GPP 5G perspective.

The ingress routers to the 5G Edge Computing Network are directly connected to 5G UPFs. The egress routers to the 5G Edge Computing Network are the routers that have a direct link to the Edge Computing servers. The servers and the egress routers are co-located. Some of those mini Edge Computing Data centers may have Virtual switches or Top of Rack switches between the egress routers and the servers. But transmission delay between the egress routers and the Edge Computing servers is very small, which is considered negligible in this document.

When multiple Edge Computing Servers attached to one App Layer Load Balancer, only the App Layer Load Balancer address is visible to the 5G Edge Computing Network. How the App Layer Load balancer manages the individual servers is out of the scope of the document.

The Edge Computer Services are registered services that need to utilize the network topology and balance among multiple mini Edge Computing Data Centers with the same ANYCAST address. Majority services are not registered 5G Edge Computing Services.

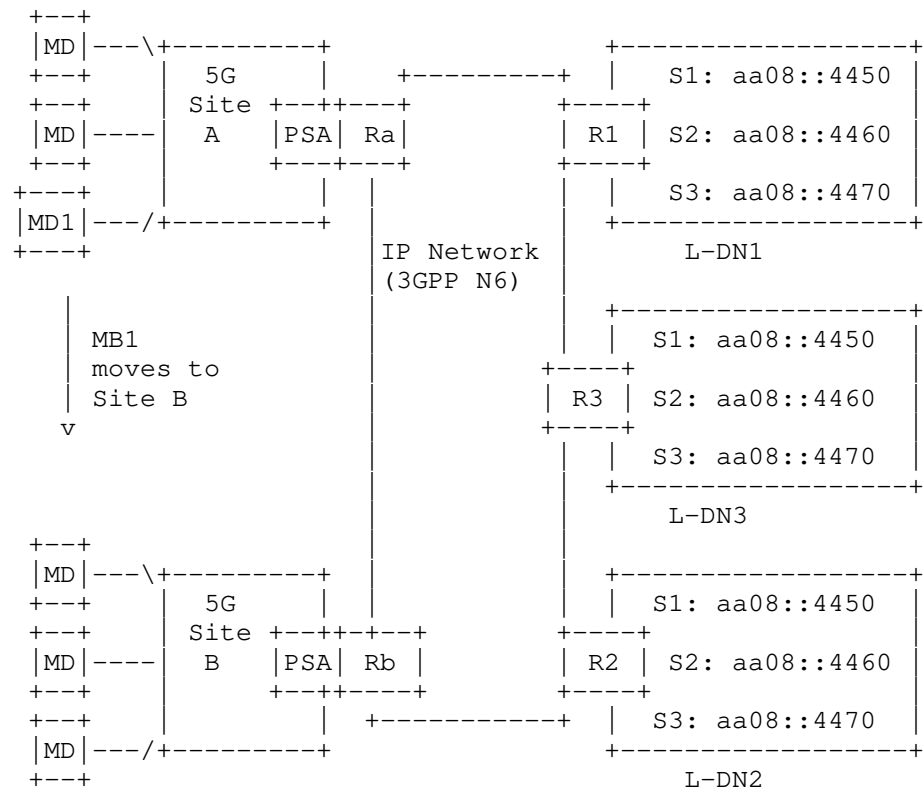


Figure 1: App Servers in different edge DCs

1.3. Problem #1: Discovery of Edge Application Server

Key Issue #1 identified by 3GPP Edge Computing Study [TR 23.748] is that one application service might be served by multiple Edge Application Servers typically deployed in different sites. These multiple Edge Application Server instances that host same content or service may use a single IP address (anycast address) or different IP addresses.

Key Issue #2 identified by 3GPP Edge Computing Study [TR 23.748] is Edge server relocation.

Application Server discovery and relocation can be achieved by running IGP/BGP routing protocols among the routers in LDN.

Increasingly, ANYCAST is used extensively by various application providers because it is possible to dynamically load balance across multiple locations of the same address based on network conditions. When multiple servers in different locations have the same IP address (ANYCAST), the routers see multiple paths to the IP address. The IGP/BGP routing protocols can inform all the nodes where the servers are and when servers move to new locations.

Application Server location selection using Anycast address leverages the proximity information present in the network routing layer and eliminates the single point of failure and bottleneck at the DNS resolvers and application layer load balancers. Another benefit of using ANYCAST address is removing the dependency on mobile devices that use their cached IP addresses instead of querying DNS when they move to a new location.

However, having multiple locations for the same ANYCAST address in the 5G Edge Computing environment can be problematic because all those edge computing Data Centers can be close in proximity. There might not be any difference in the routing cost to reach the Application Servers in different Edge DCs. The same routing cost to multiple locations can cause packets from one flow to be forwarded to different locations, which can cause service glitches.

1.4. Problem #2: sticking to original App Server

When a mobile device moves to a new location but continues the same application flow, the router connected to the new UPF might choose the App Server closer to the new location. As shown in the figure below, when the MD1 in 5G-site-A moves to the 5G-Site-B, the router directly connected to 5G PSA2 might forward the packets destined towards the S1: aa08::4450 to the server located in L-DN2 because L-DN2 has the lowest cost based on routing. This is not the desired behavior for some services, which are called Sticky Services in this document.

Even for some advanced applications with built-in mechanisms to re-sync the communications at the application layer after switching to a new location, service glitches are often experienced.

It worth noting that not all services need to be sticky. We assume only a subset of services are, and the Network is informed of the services that need to be sticky, usually by requests from application developers or controllers.

This document describes an IPv6-based network layer solution to stick the packets belonging to the same flow of a mobile device to its original App Server location after the mobile device is anchored to a new nearby UPF-PSA.

Note: for ease of description, the Edge Computing Server, Application Server, or App Server are used interchangeably throughout this document.

2. Conventions used in this document

APN6 Application aware network using IPv6. The term "Application" has very broad meanings. In this document the term "Application" refers to any applications that use ANYCAST servers in the 5G Edge Computing Environment.

A-ER: Egress Router to an Application Server, [A-ER] is used to describe the last router that the Application Server is attached. For 5G EC environment, the A-ER can be the gateway router to a (mini) Edge Computing Data Center.

Application Server: An application server is a physical or virtual server that host the software system for the application.

Application Server Location: Represent a cluster of servers at one location serving the same Application. One application may have a Layer 7 Load balancer, whose address(es) are reachable from external IP network, in front of a set of application servers. From IP network perspective, this whole group of servers are considered as the Application server at the location.

Edge Application Server: used interchangeably with Application Server throughout this document.

EC: Edge Computing

Edge Hosting Environment: An environment providing support required for Edge Application Server's execution.

NOTE: The above terminologies are the same as those used in 3GPP TR 23.758

Edge DC: Edge Data Center, which provides the Edge Computing Hosting Environment. It might be co-located with or very close to a 5G Base Station.

gNB next generation Node B

L-DN: Local Data Network

MD: Mobile Device, which is the same as the UE (User Equipment) used in 3GPP. The term "mobile device" is used instead of UE to emphasize on sticking services originated from the devices that are mobile to same server.

PSA: PDU Session Anchor (UPF)

SSC: Session and Service Continuity

UE: User Equipment. UE is same as a mobile device in this document.

UPF: User Plane Function

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

3. Stick a Flow to an ANYCAST Server

When servers attached to different egress routers are assigned with the same IP address, the routers in the LDN see multiple paths to the IP address. The Egress nodes' unicast addresses are the Next Hops (i.e., R1, R2, and R3) to reach the Edge Computing server ANYCAST address.

The routers choose the lowest cost path. [5G-EC-OSPF-EXT] and [5G-EC-BGP-EXT] describe the OSPF and BGP extension to propagate additional costs about the site where the servers are located so that the site costs can be incorporated into the path computation.

Flow sticking to one server is not the same as flow nailing down to the same server. When the network cost is significantly increased, such as the mobile device moving to a very far away location or the extreme case of link failure to the original server, another server with the same IP address is selected.

The Flow Affinity feature, which most commercial routers support today, can ensure packets belonging to one flow be forwarded along the same path to the same egress router, which then delivers the packets to the attached server.

Editor's note: for IPv6 traffic, Flow Affinity can be supported by the Local Data Network (LDN) routers forwarding the packets with the same Flow Label in the packets' IPv6 Header along the same path towards the same egress router. For IPv4 traffic, 5 tuples in the IPv4 header can be used to achieve the Flow Affinity.

When a UE moves to a different cell site, the packets from the UE might enter the 5G LDN from a different UPF. Suppose the handover to the new cell site is in the middle of a flow from the UE. In that case, the new ingress router directly connected to the new UPF needs to have the original egress router information to stick the flow from the UE to the original egress router. The original egress router is called Sticky Egress throughout this document.

4. Sticky flow for QUIC based Applications

For applications using QUIC transport protocol, ANYCAST stickiness are supported natively. During the initial handshake, QUIC servers can provide a "preferred address" (IP or IPv6 and port number), and the client can immediately migrate the connection to use that address. This was

specifically designed to support servers listening on anycast addresses, so the connection can be pinned to a unicast address specific to the server.

5. Other Solutions within a Limited Domain

This section describes some sticky flow solutions within a limited domain [RFC8799] for applications not based on QUIK.

Within a limited domain [RFC8799], mobile devices, edge servers, and network functions are under one administrative domain. Therefore, it is feasible for mobile devices to perform specific actions.

5.1. Use Case of 5G Edge Computing in a limited domain.

Some 5G Connected devices, such as drones for fighting natural disasters or robots in Industry 4.0 environments, need ultra-low latency responses from their analytic servers. To reach ultra-low latency, those analytic functions can be hosted on servers very close to radio towers.

All the functions (including networking and analytics) and devices are administrated by one operator. Network devices within the 5G LDN limited domain might be provided by different vendors, therefore needing interoperable solutions.

5.2. End Node Based Sticky Service Solution

The End-Node-based Sticky Service solution needs IPv6 mobile devices to insert the Destination Option header extracted from the packet received from the network side to the IPv6 Header of the next packet if the next packet belongs to the same flow. This action dramatically simplifies the processing at the LDN's Ingress routers.

Here are some assumptions for the End-Node based Sticky Service solution:

- The mobile devices are under the same administrative control as the Edge computing servers.
- If an Edge Computing service needs to be sticky in the 5G Edge Computing environment, the corresponding service ID is registered with the 5G Edge Computing controller. The Sticky Service ID can be the IP address (unicast or ANYCAST) of the server.

Here is the overview of the End-Node based Sticky Service solution:

- Each ANYCAST Edge Computing server either learns or is informed of the unicast Sticky Egress address (Section 3). The goal is to deliver packets belonging to one flow to the same Sticky Egress address for the ANYCAST address.
- When an Edge Computing server sends data packets back to a client (or the mobile device), it inserts the Sticky-Dst-SubTLV (described in Section 4.4) into the packets' Destination Option Header.
- The client (or the mobile device) needs to copy the Destination Option Header from the received packet to the next packet's Destination Header if the next packet belongs to the same flow as the previous packet.
- If the following conditions are true, the ingress router encapsulates the packet from the client in a tunnel whose outer destination address is set to the Sticky Egress Address extracted from the packet's Sticky-Dst-SubTLV:
 - o The destination of the packet from the client-side matches with one of the Sticky Service ACLs configured on the ingress router of the LDN,
 - o the packet header has the Destination Option present with Sticky-Dst-SubTLV.
- Else (i.e., one of the conditions above is not true), the ingress node uses its algorithm, such as the least cost as described in [5G-EC-Metrics], to select the optimal Sticky Egress address for forwarding the packet.

5.2.1. Edge Controller Based Solution.

To be added.

[Editor's note: can consider adding something along the line of the following, which is suggested by the email: say 5G/MEC control plane can tell the UE what address to use, it does NOT mean a UE will query whenever it is anchored to a new UPF. The initial query when it needs a service will return the unicast address of a server based

on all kinds of information/constraints, including the server load information talked about in draft-dunbar-idr-5g-edge-compute-app-meta-data. After that, the server won't change until new server is indeed needed (this is what "sticky service" is about, right). When a server change is indeed needed, the 5G/MEC control plane will tell the UE the new unicast address to use and tell the servers to move the corresponding application data when necessary.

]

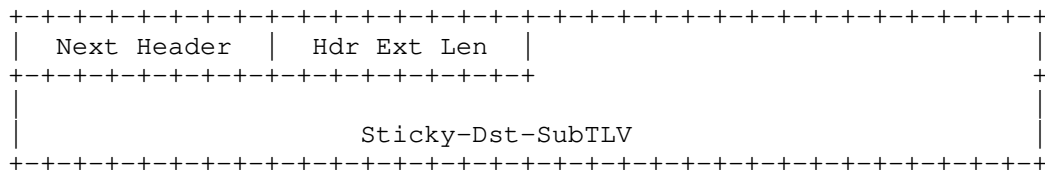
5.3. Sticky Egress Address Discovery

To an App server with ANYCAST address, the Sticky Egress address is the same as its default Gateway address.

To prevent malicious entities sending DDOS attacks to routers within 5G EC LDN, e.g., the Sticky Egress address that is encoded in the Destination option header in the packets sent back to the clients, a proxy Sticky Egress address can be encoded in the Destination option header. The proxy Sticky Egress address is only recognizable by the 5G EC LDN ingress nodes, i.e., the Ra and Rb in Figure 1, but not routable in other networks. The LDN ingress routers can translate the proxy Sticky Egress to a routable address for the Sticky Egress node after the source addresses of the packets are authenticated.

5.4. Sticky-Dst-SubTLV in Destination Extension Header

A new Sticky-Dst-SubTLV is specified as below, which can be inserted into the IPv6 Destination Options header. The IPv6 Destination Option Header is specified by [RFC8200] as having a Next Header value of 60:



Sticky-Dst-SubTLV is specified as:

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Sticky-Type |           Len | AFI |           Reserved |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Sticky Egress address (IPv4 or IPv6) for reaching the ANYCAST |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Sticky-Type = 1: indicate the Sticky Egress unicast address at encoded in the Sticky-Dst-SubTLV.

5.5. Processing at the Ingress router

- An Ingress router is configured with an ACL for filtering out the applications that need sticky service.

Note, not all applications need sticky service. Using ACL can significantly reduce the processing on the routers.

- When an Ingress router receives a packet from the 5G side that matches the ACL, the Ingress router extracts the Sticky-Dst-SubTLV from the packet IPv6 header if the field exists in the packet header.
- Encapsulate the packet with the tunnel type that are supported by the original Sticky Egress node, using the extracted Sticky Egress address in the destination field of the outer Header, and forward the packet.

Note: if the proxy Sticky Egress address is encoded in the Sticky-Dst-SubTLV, the ingress router needs to translate the proxy Sticky Egress address to a routable address.

If none of the above conditions are met, the ingress router uses its algorithm to select the optimal Sticky Egress node to forward the packet.

6. Tunnel based Sticky Service Solution

For environments that mobile devices cannot change their processing behavior as described in Section 4, a Tunnel based

Sticky Service solution can be used. This solution does not depend on mobile device's behavior. However, this solution does require ingress routers to filter out the registered sticky services and might need some level of assistance from the LDN network controller.

6.1. Desired functions by the Network Controller

6.2. Ingress and Egress Routers Processing Behavior

The solution assumes that both ingress routers and egress routers support at least one type of tunnel and are configured with ACLs to filter out packets whose destination or source addresses match with the Sticky Service Identifier. The solution also assumes there are only limited number of Sticky Services to be supported.

An ingress router needs to build a Sticky-Service-Table, with the following minimum attributes. The Sticky-Service-Table is initialized to be empty.

- Sticky Service ID
- Flow Label
- Sticky Egress address
- Timer

Editor's Note:

When a mobile device moves from one 5G Site to another, the same mobile device will have a new IP address. "Flow Label + Sticky Service ID" stays the same when a mobile device is anchored to a new PSA. Therefore, this solution uses "Flow Label + Sticky Service ID" to identify a sticky flow. Since the chance of different mobile devices sending packets to the same ANYCAST address using the same Flow Label is very low, it is with high probability that "Flow Label + Sticky Service ID" can uniquely identify a flow. When multiple mobile devices using the same Flow Label sending packets to the same ANYCAST address, the solution described in this section will stick the flows to the same ANYCAST server attached to the Sticky Egress router. This behavior doesn't cause any harm.

Each entry in the Sticky-Service-Table has a Timer because a sticky service is no longer sticky if there are no packets of the same flow destined towards the service ID for a period of time. The Timer should be larger than a typical TCP session Timeout value. An entry is automatically removed from the Sticky-Service-Table when its timer expires.

Note: since there are only small number of Sticky services, the Sticky-Service-Table is not very large.

When an ingress router receives a packet from a mobile device matching with one of the Sticky Service ACLs and there is no entry in the Sticky-Service-Table matching the Flow Label and the Sticky Service ID, the ingress router considers the packet to be the first packet of the flow. There is no need to sticking the packet to any location. The ingress router uses its own algorithm to select the optimal egress node as the Sticky Egress address for the ANYCAST address, encapsulates the packet with a tunnel that is supported by the egress node. The tunnel's destination address is set to the egress node address.

When an egress router receives a packet from an attached host with the packet's source address matching with one of the Sticky Service IDs, the egress router encapsulates the packet with a tunnel that is supported by the ingress router and the tunnel's destination address is set to the ingress router address. An Egress router learns the ingress router address for a mobile device IP address via BGP UPDATE messages.

When an ingress router receives a packet in a tunnel from any egress router and the packet's source address matches with a Sticky Service ID, the egress router address is set as the Sticky Egress address for the Sticky Service ID. The ingress router adds the entry of "Sticky-Service-ID + Flow Label + the associated Sticky Egress address + Timer" to the Sticky-Service-Table if the entry doesn't exist yet in the table. If the entry exists, the ingress router refreshes the Timer of the entry in the table.

When the ingress router receives the subsequent packets of a flow from the 5G side matching with an Sticky Service ID and the Sticky-Service ID exists in the Sticky-Service-Table, the ingress router uses the Sticky Egress address found in the Sticky-Service-Table to encapsulate the packet and refresh the Timer of the entry. If the Sticky-Service ID doesn't exist in the table, the ingress router considers the packet as the first packet of a flow.

The subsequent sections describe how ingress nodes prorogate their Sticky-Service-Table to their neighboring ingress nodes. The propagation is for neighboring ingress nodes to be informed of the Sticky Egress address to a sticky service if a mobile device moves to a new neighboring 5G site resulting in anchoring to a new ingress node.

6.3. A Solution without the Communication with 5G system.

When a mobile device moves to a very far away 5G site, say a different geographic region, the benefit of sticking to the original ANYCAST server is out weighted by network delay. Then, there is no point sending packets to the Sticky Egress node if the ingress router very far away. Therefore, it is necessary for each ingress router to have a group of neighboring ingress routers that are not too far away from the potential Sticky Egress nodes selected by the ingress router. This group of ingress routers is called the Neighboring Ingress Group. Each ingress router can either automatically discover its Neighboring Ingress Group by routing protocols or is configured by its controller. It is out of the scope of this document on how ingress nodes discover its Neighboring Ingress Group.

Each ingress node needs to periodically advertise its Sticky-Service-Table to the routers within its Neighboring Ingress Group.

Upon receiving the Sticky-Service-Table from routers in its Neighboring Ingress Group, each ingress router merges the entries from the received Sticky-Service-Table to its own.

The ingress and the egress nodes perform the same actions as described in Section 5.1.

6.4. A Solution that depends on the communication with 5G system

In this scenario, there is communication with 5G System and network get notified by a mobile device is anchored to a new PSA.

When a mobile device is re-anchoring from PSA1 to PSA2, 5GC EC management system sends a notification to the router that is directly connected to PSA1. The notification includes the address of the new PSA that the mobile device is to be anchored, i.e. the PSA2, and the mobile device's new IP address.

In this scenario, the Sticky Service can be uniquely identified by "Sticky Service ID" + "mobile device address". the Sticky-Service-Table should include the following attributes:

- Sticky Service ID
- mobile device address
- Sticky Egress address
- Timer

Upon receiving the notification from the 5G EC management system, the ingress router (i.e. the one directly connected to the old PSA) sends the specific entry of the Sticky-Service Table, i.e. "Sticky Service ID" + mobile device address + Sticky Egress + Timer to the router directly connected to the new PSA.

Upon receiving the entry, the ingress router merges the entry into its own Sticky-Service-Table.

The ingress and egress router processing are the same as described in Section 5.1 except a flow is now uniquely identified by the "Sticky Service ID" + "mobile device address" instead of "Sticky Service ID" + "Flow Label".

7. Expanding APN6 for Sticky Service information

The Application-aware ID and Service-Para Option described [APN6] can be expanded to include the sticky service information.

7.1. Sticky Service ID encoded in the Application-aware ID

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Sticky Level | StickyServiceID | Reserved      | Flow ID      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Sticky Level: represent how important for an application to stick to its ANYCAST servers. Some applications may prefer one flow sticking to the original ANYCAST server, but not required. Some applications may require the stickiness.

StickyServiceID: the ANYCAST address of the application servers.

The Reserved field can be used for future to identifier the 5G access domain for the flow.

Flow ID: the identifier for the flow that needs to stick to a specific ANYCAST server.

7.2. Sticky Service Sub-TLV encoded in APN6 Service-para option

The Sticky-Dst-SubTLV described in the Section 4.2 of this document can be included in the Service-Para Sub-TLVs field.

8. Manageability Considerations

To be added.

9. Security Considerations

To be added.

10. IANA Considerations

To be added.

11. References

11.1. Normative References

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

[RFC4364] E. rosen, Y. Rekhter, "BGP/MPLS IP Virtual Private networks (VPNs)", Feb 2006.

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

[RFC8200] s. Deering R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", July 2017

11.2. Informative References

- [3GPP-EdgeComputing] 3GPP TR 23.748, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Study on enhancement of support for Edge Computing in 5G Core network (5GC)", Release 17 work in progress, Aug 2020.
- [5G-EC-Metrics] L. Dunbar, H. Song, J. Kaippallimalil, "IP Layer Metrics for 5G Edge Computing Service", draft-dunbar-ippm-5g-edge-compute-ip-layer-metrics-00, work-in-progress, Oct 2020.
- [5G-EC-OSPF-EXT] L. Dunbar, H.Chen, A. Wang, "OSPF extension for 5G Edge Computing Service", draft-dunbar-lsr-5g-edge-compute-ospf-ext-05, work-in-progress, March 2021.
- [5G-EC-BGP-EXT] L. Dunbar, K. Majumdar, H. Wang, "BGP NLRI App Meta Data for 5G Edge Computing Service", draft-dunbar-idr-5g-edge-compute-app-meta-data-02, work-in-progress, March 2021.
- [APN6] Z. Li, et al, "Application-aware IPv6 Networking (APN6) Encapsulation", draft-li-6man-app-aware-ipv6-network-03, work-in-progress, Feb 2021.
- [RFC5521] P. Mohapatra, E. Rosen, "The BGP Encapsulation Subsequent Address Family Identifier (SAFI) and the BGP Tunnel Encapsulation Attribute", April 2009.
- [BGP-SDWAN-Port] L. Dunbar, H. Wang, W. Hao, "BGP Extension for SDWAN Overlay Networks", draft-dunbar-idr-bgp-sdwan-overlay-ext-03, work-in-progress, Nov 2018.

[SDWAN-EDGE-Discovery] L. Dunbar, S. Hares, R. Raszuk, K. Majumdar, "BGP UPDATE for SDWAN Edge Discovery", draft-dunbar-idr-sdwan-edge-discovery-00, work-in-progress, July 2020.

[Tunnel-Encap] E. Rosen, et al "The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-encaps-10, Aug 2018.

12. Acknowledgments

Acknowledgements to Gyan Mishra, Jeffrey Zhang, Joel Halpern, Ron Bonica, Donald Eastlake, and Eduard Vasilenko for their review and contributions.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Linda Dunbar
Futurewei
Email: ldunbar@futurewei.com

John Kaippallimalil
Futurewei
Email: john.kaippallimalil@futurewei.com

IPv6 Maintenance Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 24, 2021

X. Geng
M. Chen
F. Yang
Huawei Technologies
February 20, 2021

SRH Extension for Redundancy Protection
draft-geng-6man-redundancy-protection-srh-00

Abstract

Redundancy protection is a method of service protection by sending copies of the same packets of one flow over multiple paths, which includes packet replication, elimination and ordering. This document defines SRv6 header (SRH) extensions to support redundancy protection.

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 .

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 August 24, 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
2. Terminology and Conventions	2
3. Redundancy Protection over SRv6 Scenario	3
4. SRH Extension for Redundancy Protection	3
5. IANA Considerations	4
6. Security Considerations	4
7. Acknowledgements	4
8. Normative References	4
Authors' Addresses	5

1. Introduction

Redundancy protection is a method of providing 1+1 protection by sending copies of the same packets of one flow over multiple paths, which includes packet replication, elimination and ordering. This document defines SRv6 header (SRH) extensions to support redundancy protection.

2. Terminology and Conventions

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].

Redundancy Node: the start point of redundancy protection, which is a network device that could implement packet replication.

Merging Node: the end point of redundancy protection, which is a network node that could implement packet elimination and ordering (optionally).

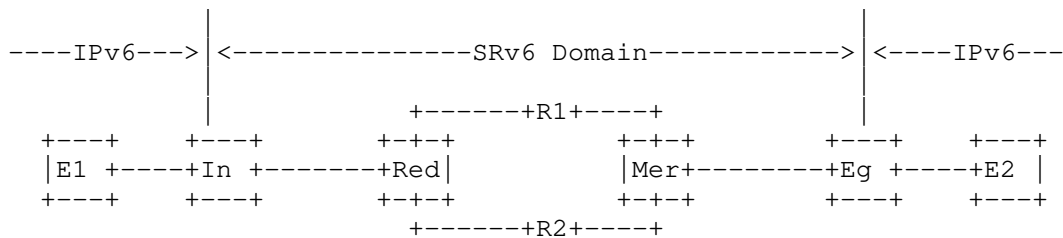
Flow Identification: information in SR data service to indicate one flow.

Sequence Number: information in SR data service to indicate the packet sequence of one flow.

Editor's Note: Similar mechanism is defined as "Service Protection" in the [RFC8655]. In this document, we define a new term "Redundancy Protection" to distinguish with other service protection method. Some of the terms are the similar as [RFC8655].

3. Redundancy Protection over SRv6 Scenario

The figure shows how to provide redundancy protection over SRv6.



As the figure shows, an IPv6 flow is sent out from the end station E1. The packet of the flow is encapsulated in an outer IPv6+SRH header in the Ingress(In) and transported through an SRv6 domain. In the Egress(Eg), the outer IPv6+SRH header of packet is popped, and the packet is sent to the destination E2.

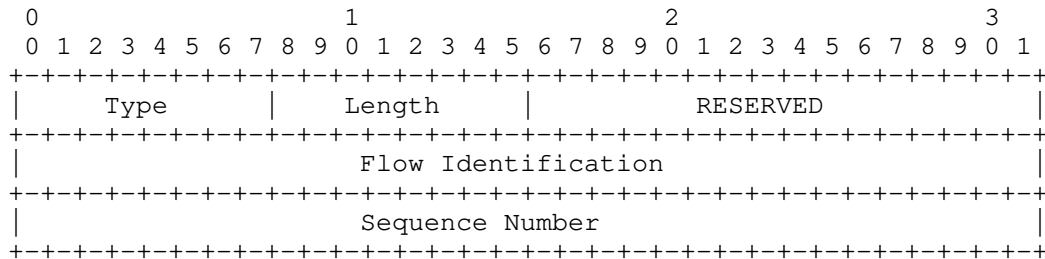
The process of redundancy protection is as follows: 1) The flow is replicated in Red (Redundancy Node); 2) Two replicated flows go through different paths till Mer (Merging Node); When there is any failures happened in one the path, the service continues to deliver through the other path without break; 3) The first received packet of the flow is transmitted from Mer (Merging Node) to Eg(Egress), and the redundant packets are eliminated; 4) Sometimes, the packet will arrive out of order because of redundancy protection, the function of reordering may be necessary in the Merging Node.

This document defines Flow Identification and Sequence Number in Segment Routing Header(SRH) as an extension of [RFC8754] to support redundancy protection.

Flow Identification is used to distinguish flows, and Sequence Number is used to distinguish packets in the same flow when doing packet merging and ordering.

4. SRH Extension for Redundancy Protection

Flow Identification and Sequence Number could be defined in SRH optional TLV.



where:

- o Type: 8 bits, indicates the use of redundancy protection, to be assigned by IANA.
- o Length: 8 bits.
- o Reserved: 16 bits. MUST be 0 on transmission and ignored on receipt.
- o Flow Identification: 32 bits, which is used for identifying the redundant protection flow.
- o Sequence Number: 32 bits, which is used for indicating sequence number of the redundant protection flow.

5. IANA Considerations

This document requires registration of a specific type of TLV used for redundancy protection in "Segment Routing Header TLVs" registry.

6. Security Considerations

TBD

7. Acknowledgements

Thanks for the valuable comments from James Guichard and Andrew Malis.

8. Normative References

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

- [RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas,
"Deterministic Networking Architecture", RFC 8655,
DOI 10.17487/RFC8655, October 2019,
<<https://www.rfc-editor.org/info/rfc8655>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J.,
Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header
(SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020,
<<https://www.rfc-editor.org/info/rfc8754>>.

Authors' Addresses

Xuesong Geng
Huawei Technologies
Beijing
China

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen
Huawei Technologies
Beijing
China

Email: mach.chen@huawei.com

Fan Yang
Huawei Technologies
Beijing
China

Email: shirley.yangfan@huawei.com

IPv6 maintenance Working Group (6man)
Internet-Draft
Updates: 4291, 4193, 8190 (if approved)
Intended status: Standards Track
Expires: July 9, 2021

F. Gont
SI6 Networks
January 5, 2021

Scope of Unique Local IPv6 Unicast Addresses
draft-gont-6man-ipv6-ula-scope-00

Abstract

Unique Local IPv6 Unicast Addresses (ULAs) are formally part of the IPv6 Global Unicast address space. However, the semantics of ULAs clearly contradict the definition of "global scope". This document discusses the why the terminology employed for the specification of ULAs is problematic, along with some practical consequences of the current specification of ULAs. Finally, it formally updates RFC4291 and RFC4193 such that the scope of ULAs is defined as "local".

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 July 9, 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
2. What Does 'Global Scope' mean?	2
3. Scope of Unique Local IPv6 Unicast Addresses	3
4. Problems with the Definition of the ULA Scope	4
5. Practical Consequences	4
5.1. Address Attributes in Programming Languages	5
6. Specification Updates	5
7. IANA Considerations	6
8. Security Considerations	7
9. Acknowledgements	7
10. References	7
10.1. Normative References	7
10.2. Informative References	8
Author's Address	8

1. Introduction

Unique Local IPv6 Unicast Addresses (commonly referred to as "ULAs" [RFC4193] are formally part of the IPv6 Global Unicast address space. However, the semantics of ULAs clearly contradict the definition of "global scope" [RFC4007].

This document discussed the specification of ULAs and, in particular, of their associated scope. Additionally, it discusses how the semantics of ULAs contradicts their formal address scope along with some and practical consequences of this problematic definition. Finally, this document formally updates RFC4193 and RFC4291, such that ULAs are defined to have "local scope" (larger than link-local, and smaller than "global").

The problematic definition of ULAs was initially encountered when analyzing IPv6 address properties while working on [I-D.gont-v6ops-ipv6-addressing-considerations]. The issue became fully-evident from discussions with Brian Carpenter, both off-list and on-list [v6ops-thread].

2. What Does 'Global Scope' mean?

[RFC4007] defines the scope of an address as:

"[the] topological span within which the address may be used as a unique identifier for an interface or set of interfaces"

And defines the "global scope" to be used for:

"uniquely identifying interfaces anywhere in the Internet"

3. Scope of Unique Local IPv6 Unicast Addresses

[RFC4193] formally specifies Unique Local IPv6 Unicast Addresses. [RFC4193] did not formally update [RFC3513], the current IPv6 Addressing Architecture at the time [RFC4193] was published. Therefore, ULAs were specified as a different address type, but rather as part of the Global Unicast address space.

[RFC3513] was eventually obsoleted by [RFC4291] (current revision of the IPv6 Addressing Architecture), but still did not formally accommodate ULAs into the IPv6 Addressing Architecture. For instance, Section 2.4 of [RFC4291] notes that the type of an IPv6 address is identified by the high-order bits of the address, as follows:

Address type	Binary prefix	IPv6 notation	Section
-----	-----	-----	-----
Unspecified	00...0 (128 bits)	::/128	2.5.2
Loopback	00...1 (128 bits)	::1/128	2.5.3
Multicast	11111111	FF00::/8	2.7
Link-Local unicast	1111111010	FE80::/10	2.5.6
Global Unicast	(everything else)		

and subsequently notes that:

"Future specifications may redefine one or more sub-ranges of the Global Unicast space for other purposes, but unless and until that happens, implementations must treat all addresses that do not start with any of the above-listed prefixes as Global Unicast addresses."

Therefore, ULAs still formally belong to the Global Unicast address space.

Additionally, Section 3.3 of [RFC4193] (the specification of Unique Local IPv6 Unicast Addresses) defines the scope of ULAs as:

"By default, the scope of these addresses is global. That is, they are not limited by ambiguity like the site-local addresses defined in [ADDARCH]. Rather, these prefixes are globally unique, and as such, their applicability is greater than site-local addresses."

4. Problems with the Definition of the ULA Scope

Section 3.3 of [RFC4193] (the specification of Unique Local IPv6 Unicast Addresses) defines the scope of ULAs as:

"By default, the scope of these addresses is global. That is, they are not limited by ambiguity like the site-local addresses defined in [ADDARCH]. Rather, these prefixes are globally unique, and as such, their applicability is greater than site-local addresses. Their limitation is in the routability of the prefixes, which is limited to a site and any explicit routing agreements with other sites to propagate them (also see Section 4.1). Also, unlike site-locals, a site may have more than one of these prefixes and use them at the same time."

However, there is a problem in this analysis: ULA prefixes have a finite probability of being globally unique. For instance, Section 3.2.3 of [RFC4193] computes the probability of collisions *when inter-connecting a limited number of networks employing ULAs*. As such, based on the definition of "scope" and "global scope" (see Section 2), ULAs cannot possibly have a "global scope" -- their scope is certainly smaller than "global". And this non-global scope does limit the global routability of ULAs since, in principle, an address cannot be routed outside of its associated zone.

The only ULAs that could possibly have "global scope" are the so-called ULA-C [I-D.ietf-ipv6-ula-central], that have so far *not* been formally specified.

It should be noted that the non-global scope of ULAs does not preclude their usage for e.g. inter-site Virtual Private Networks (VPN), as discussed in Section 4.7 of [RFC4193]. For example, the private address space specified in [RFC1918] for IPv4 networks has non-global scope, but still is regularly used for inter-site VPNs. ULAs having a non-global scope simply means that while allocating "Global IDs" from a Pseudo-Random Number Generator (PRNG) reduces the probability of collisions of Global IDs *when a limited number of networks employing ULAs are interconnected*, ULA prefixes cannot be expected to be globally unique.

"Global scope" would imply that all ULA prefixes in use by any networks, whether interconnected or not, are unique.

5. Practical Consequences

5.1. Address Attributes in Programming Languages

Python's `ipaddress` library [Python-ipaddr] defines '`IPv6Address`' objects that have a number of attributes, including:

- o '`True`' if the address is allocated for private networks.
- o '`True`' if the address is allocated for public networks.

For ULAs, the `is_private` attribute is '`True`', while the `is_global` attribute is '`False`'. This contradicts the definition of ULAs as having "global scope" [RFC4291] [RFC4193], but is in line with the specification update performed by this document (see Section 6).

6. Specification Updates

The ultimate goal is to employ coherent terminology and definitions throughout the relevant protocol specifications. Probably the only option to achieve this goal is update the definition of ULAs as having "local scope", with "local scope" defined as "larger than link-local, and smaller than global" (based on ULAs being defined as "local addresses").

- o [TBD: Analyze possible implications on Default Address Selection for Internet Protocol Version 6 (IPv6) [RFC6724].]

The following table from Section 2.4 of [RFC4291]:

---- cut here ----			
Address type	Binary prefix	IPv6 notation	Section
-----	-----	-----	-----
Unspecified	00...0 (128 bits)	::/128	2.5.2
Loopback	00...1 (128 bits)	::1/128	2.5.3
Multicast	11111111	FF00::/8	2.7
Link-Local unicast	1111111010	FE80::/10	2.5.6
Global Unicast	(everything else)		
---- cut here ----			

is replaced with:

---- cut here ----

Address type	Binary prefix	IPv6 notation	Reference
-----	-----	-----	-----
Unspecified	00...0 (128 bits)	::/128	Sec. 2.5.2
Loopback	00...1 (128 bits)	::1/128	Sec. 2.5.3
Unique Local unicast	1111110	FC00::/7	[RFC4193]
Multicast	11111111	FF00::/8	Sec. 2.7
Link-Local unicast	1111111010	FE80::/10	Sec. 2.5.6
Global Unicast	(everything else)		

---- cut here ----

The following text from Section 3.3 of [RFC4193]:

---- cut here ----

By default, the scope of these addresses is global. That is, they are not limited by ambiguity like the site-local addresses defined in [ADDARCH]. Rather, these prefixes are globally unique, and as such, their applicability is greater than site-local addresses. Their limitation is in the routability of the prefixes, which is limited to a site and any explicit routing agreements with other sites to propagate them (also see Section 4.1). Also, unlike site-locals, a site may have more than one of these prefixes and use them at the same time.

---- cut here ----

is replaced with:

---- cut here ----

The scope of these addresses is 'local', defined to be 'larger than link-local, but smaller than global'. Their limitation is in the routability of the prefixes, generally limited by any explicit routing agreements with other autonomous systems (ASes) to propagate them, and normally limited by the Default-Free Zone (DFZ) (also see Section 4.1).

---- cut here ----

7. IANA Considerations

The IANA is instructed to update the "IANA IPv6 Special-Purpose Address Registry" [IANA-ADDR-REG] by adding a "[RFCXXXX]" to the "RFC" column corresponding to the "fc00::/7" address block.

Additionally, the following footnote:

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

must be replaced with:

[4] See [RFC4193] for more details on the routability of Unique-Local addresses, and [RFCXXXX] for details on the scope of Unique-Local addresses.

NOTE: [RFCXXXX] represents the RFC number assigned by the RFC Editor upon publication of this document as an RFC.

8. Security Considerations

This document does not introduce any new security considerations.

9. Acknowledgements

Fernando Gont would like to thank Brian Carpenter and Bob Hinden, for providing valuable comments on earlier versions of this document.

Fernando Gont would like to thank Brian Carpenter for his end-less help, and for the discussion that eventually led to this document.

10. References

10.1. Normative References

- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.
- [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005, <<https://www.rfc-editor.org/info/rfc4007>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC8190] Bonica, R., Cotton, M., Haberman, B., and L. Vegoda, "Updates to the Special-Purpose IP Address Registries", BCP 153, RFC 8190, DOI 10.17487/RFC8190, June 2017, <<https://www.rfc-editor.org/info/rfc8190>>.

10.2. Informative References

- [I-D.gont-v6ops-ipv6-addressing-considerations]
Gont, F. and G. Gont, "IPv6 Addressing Considerations",
draft-gont-v6ops-ipv6-addressing-considerations-00 (work
in progress), December 2020.
- [I-D.ietf-ipv6-ula-central]
Hinden, R., "Centrally Assigned Unique Local IPv6 Unicast
Addresses", draft-ietf-ipv6-ula-central-02 (work in
progress), June 2007.
- [IANA-ADDR-REG]
IANA, "IANA IPv6 Special-Purpose Address Registry",
<<https://www.iana.org/assignments/iana-ipv6-special-registry/iana-ipv6-special-registry.xhtml>>.
- [Python-ipaddr]
Python 3.3, "ipaddress -- IPv4/IPv6 manipulation library",
<<https://docs.python.org/3/library/ipaddress.html>>.
- [RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6
(IPv6) Addressing Architecture", RFC 3513,
DOI 10.17487/RFC3513, April 2003,
<<https://www.rfc-editor.org/info/rfc3513>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown,
"Default Address Selection for Internet Protocol Version 6
(IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012,
<<https://www.rfc-editor.org/info/rfc6724>>.
- [v6ops-thread]
v6ops wg, "[v6ops] I-D Action: draft-gont-v6ops-ipv6-
addressing-considerations-00.txt", email thread on the
v6ops wg mailing-list, 2020,
<<https://mailarchive.ietf.org/arch/msg/v6ops/b7r35HgOb-6dfxsDoW8c4FtGnZo//>>>.

Author's Address

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

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

Network Working Group
Internet-Draft
Updates: 8200 (if approved)
Intended status: Standards Track
Expires: 4 December 2021

R. Hinden
Check Point Software
G. Fairhurst
University of Aberdeen
2 June 2021

IPv6 Hop-by-Hop Options Processing Procedures
draft-hinden-6man-hbh-processing-01

Abstract

This document specifies procedures for how IPv6 Hop-by-Hop options are processed. It modifies the procedures specified in the IPv6 Protocol Specification (RFC8200) to make processing of IPv6 Hop-by-Hop options practical with the goal of making IPv6 Hop-by-Hop options useful to deploy and use in the Internet. When published, this document updates RFC8200.

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 4 December 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
2. Requirements Language	2
3. Terminology	2
4. Background	3
5. Hop-by-Hop Header Processing Procedures	5
5.1. Hop-by-Hop Options Per Packet	6
5.2. Hop-by-Hop Headers Processing	6
5.3. Router Alert Option	7
5.4. Configuration	8
6. New Hop-by-Hop Options	9
7. IANA Considerations	9
8. Security Considerations	10
9. Acknowledgments	11
10. Change log [RFC Editor: Please remove]	11
11. Normative References	11
12. Informative References	12
Authors' Addresses	12

1. Introduction

This document specifies procedures for how IPv6 Hop-by-Hop options are processed. It modifies the procedures specified in the IPv6 Protocol Specification (RFC8200) to make processing of IPv6 Hop-by-Hop options practical with the goal of making IPv6 Hop-by-Hop options useful to deploy and use in the Internet.

When published this document updates [RFC8200].

The current list of defined Hop-by-Hop options can be found at [IANA-HBH].

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.

3. Terminology

This document uses the following loosely defined terms:

- * Forwarding Plane: IPv6 hosts exchange user data through the forwarding plane. User data is processed by its recipient (i.e., an IPv6 host). User data can traverse intermediate nodes (i.e., routers) between its source and its destination. These intermediate nodes process metadata contained in packet headers. However, they do not process information contained in packet payloads.
- * Control Plane: IPv6 routers exchange management and routing information with controllers. They also exchange routing information with one another. Management and routing information is processed by its recipient (i.e., an IPv6 router or controller). Management and control information can traverse intermediate nodes (i.e., routers) between its source and its destination. These intermediate nodes process metadata contained in packet headers. However, they do not process information contained in packet payloads. So, from their perspective, this information is user data.
- * Fast Path: A path through a router that is optimized for forwarding packets without processing their payloads. The Fast Path may be supported by Application Specific Integrated Circuits (ASICs), Network Processor (NP), or other special purpose hardware. This is the usual processing path within a router taken by the forwarding plane.
- * Slow Path: A path through a router that is capable of general purpose processing and is not optimized for any particular function. This processing path is used for packets that require special processing or differ from assumptions made in Fast Path heuristics, or to process router control protocols used by the control plane.

NOTE: This distinct separation between hardware and software processing from [RFC6398] does not apply to all router architectures. However, a router that performs all or most processing in software might still incur more processing cost when providing special processing (aka Slow Path).

[RFC6192] is an example of how designs can separate control plane (Slow Path) and forwarding plane (Fast Path) functions.

4. Background

In the first version of the IPv6 specification, Hop-by-Hop options were required to be processed by all nodes: routers and hosts. This proved to not be practical in high speed routers due to several factors, including:

- * Inability to process the hop-by-hop options at wire speed on the Fast Path.
- * Hop-by-Hop options would be sent to the Slow Path. This could degrade the a router's performance and it's ability to process important control traffic.
- * A mechanism that forces packets from any source to the routers "Slow Path" could be exploited as a Denial of Service attack against the router.
- * Packets could contain multiple Hop-by-Hop options making the previous issues worse by increasing the complexity required to process them.

When the IPv6 Specification was updated and published in July 2017 as [RFC8200], the procedures relating to hop-by-hop options were as follows:

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

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

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

The changes meant that an implementation complied with the IPv6 specification even if it did not process hop-by-hop options, and that it was expected that routers would add configuration information to control which hop-by-hop options they would process.

Unfortunately, this did not improve the processing of Hop-by-Hop options and did not significantly improve deployment and use in the Internet. Essentially, it only documented how they were being used in the Internet at the time RFC8200 was published.

The main issues remain:

- * Routers are commonly configured to drop transit packets containing hop-by-hop options that would have be processed in the Slow Path. This behavior is seen as protecting against a denial of service attack on the router. This is discussed in [I-D.ietf-v6ops-ipv6-ehs-packet-drops].
- * Allowing multiple hop-by-hop options in a single packet makes it even more expensive in router resources to process these packets. It adds complexity to the number of permutations that might need to be processed.
- * Any mechanism that can be used to force packets into the router's Slow Path can be exploited as a denial of service attack on a transit router by saturating the resources needed for router management protocols (e.g., routing protocols, network management protocols, etc.) that may cause the router to fail. This issue for the Router Alert option, which intentionally places packets on the Slow Path, is discussed in [RFC6398]. Section 3 of that RFC includes a good summary:

"In a nutshell, the IP Router Alert Option does not provide a convenient universal mechanism to accurately and reliably distinguish between IP Router Alert packets of interest and unwanted IP Router Alert packets. This, in turn, creates a security concern when the IP Router Alert Option is used, because, short of appropriate router-implementation-specific mechanisms, the router Slow Path is at risk of being flooded by unwanted traffic."

There has been research that discussed the general problem with dropping packets containing IPv6 extension headers, including the Hop-by-Hop Options header. For example [Hendriks] states that "dropping all packets with Extension Headers, is a bad practice", and that "The share of traffic containing more than one EH however, is very small. For the design of hardware able to handle the dynamic nature of EHs, we therefore recommend to support at least one EH".

This document defines a set of procedures for the hop-by-hop option header that make the processing of hop-by-hop options practical in modern transit routers.

5. Hop-by-Hop Header Processing Procedures

This section describes several changes to [RFC8200].

5.1. Hop-by-Hop Options Per Packet

The Hop-by-Hop Option Header as defined in Section 4.3 of [RFC8200] is identified by a Next Header value of 0 in the IPv6 header. Section 4.1 of [RFC8200] requires a Hop-by-Hop Options header to appear immediately after the IPv6 header. [RFC8200] also requires that a Hop-by-Hop Options header can only appear once in a packet.

The Hop-by-Hop Options Header as defined in [RFC8200] can contain one or more Hop-by-Hop options. This document updates [RFC8200] that a node **MUST** process the first Option in the Hop-by-Hop Header in the Fast Path and **MAY** process additional Hop-by-Hop Options if configured to do so. The motivation for this change is to simplify the processing of Hop-by-Hop options in the Fast Path.

Nodes creating packets with a Hop-by-Hop option headers **SHOULD** include a single Hop-by-Hop Option in the packet and **MAY** include more based on local configuration.

If there are more than one Hop-by-Hop options in the Hop-by-Hop Options header, the node **MAY** skip the rest of the options without having to examine these options using the "Hdr Ext Len" field in the Hop-by-Hop Options header. This field specifies the length of the Option Header in 8-octet units. The additional options do not need to be processed or verified.

5.2. Hop-by-Hop Headers Processing

Nodes that implement a differentiation between a Fast Path and a Slow Path **MUST** process all (with one exception noted below) Hop-by-Hop options in the Fast Path. The one exception to this is the Router Alert Option [RFC2711]. See Section 5.3 for discussion of the Router Alert.

If the node can not process an option in the Fast Path, it **MUST** behave as if it does not recognize the Option Type (as described in the next paragraph).

Section 4.2 of [RFC8200] defines the Option Type identifiers as internally encoded such that their highest-order 2 bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type. The text is:

- 00 - skip over this option and continue processing the header.
- 01 - discard the packet.
- 10 - discard the packet and, regardless of whether or not the packet's Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.
- 11 - discard the packet and, only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

This document modifies this behaviour for the "10" and "11" values that the node MAY send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type. The modified text for "10" and "11" values is:

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

The motivation for this change is to loosen the requirement to send ICMPv6 Parameter Problem messages to simplify what the router needs to do in the Fast Path when it does not recognize the Option Type.

When an ICMP Parameter Problem, Code 2, message is delivered to the source, the source can become aware that at least one node on the path has failed to recognize the option.

5.3. Router Alert Option

The Router Alert option [RFC2711] purpose is to tell the node that the packet needs additional processing on the Slow Path.

The Router Alert option includes a two octet Value field that describes the protocol that is carried in the packet. The current values can be found in the IANA Router Alert Value registry [IANA-RA].

DISCUSSION

The Router Alert Option is a problem since it's function is to do what this specification is proposing to eliminate, that is, process the packet in the Slow Path. One approach would be to deprecate it as it's usage appears to be limited and packets containing Hop-by-Hop options are frequently dropped. Deprecation would allow current implementations to continue and it's use could be phased out over time.

The authors current thinking is that the Router Alert function may have reasonable potential use for new functions that have to be processed in the Slow Path. We think that keeping it as the single exception for Slow Path processing with the following restrictions is a reasonable compromise to allow future flexibility. These are compatible with Section 5 of [RFC6398].

A Fast Path implementation SHOULD verify that a Router Alert contains a protocol, as indicated by the Value field in the Router Alert option, that is configured as a protocol of interest to that router. A verified packet SHOULD be sent on the Slow Path for processing [RFC6398]. Otherwise, the router implementation SHOULD forward within the Fast Path (subject to all normal policies and forwarding rules). As specified in [RFC2711] the top two bits of Option Type for the Router Alert option are always set to "00" indicating the node should skip over this option and continue processing the header in this case.

Implementations of the IP Router Alert Option SHOULD offer the configuration option to simply ignore the presence of "IP Router Alert" in IPv4 and IPv6 packets" [RFC6398].

A node that is configured to process a Router Alert option using the Slow Path MUST protect itself from infrastructure attack that could result from processing on the Slow Path. This might include some combination of access control list to only permit from trusted nodes, rate limiting of processing, or other methods [RFC6398].

5.4. Configuration

Section 4 of [RFC8200] allows for a router to control it's processing of IPv6 Hop-by-Hop options by local configuration. The text is:

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.

A possible approach to implementing this is to maintain a lookup table based on Option Type of the IPv6 options that are supported in the Fast Path. This would allow for a node to quickly determine if an option is supported and can be processed. If the option is not supported, then the node processes it as described in Section 5.2 of this document.

A node configured not to process HBH options, MUST drop the packet if the top two bits of the Option Type field of the first HBH option is non-zero.

The actions of the lookup table SHOULD be configurable by the operator of the router.

6. New Hop-by-Hop Options

Any new IPv6 Hop-by-Hop option designed in the future should be designed to be processed in the Fast Path. New options MUST NOT be defined that require Slow Path processing. New Hop-by-Hop options SHOULD have the following characteristics:

- * Straight forward to process. That is, they should be designed to keep the time to process low.
- * Fixed size in 8-octet units. Specifically any new Hop-by-Hop options should not be variable size that could extend beyond what can be executed in the Fast Path.

Any new Hop-by-Hop option that is standardized that does not meet these criteria needs to explain in detail in its specification why this can not be accomplished and that there is a reasonable expectation that it can be proceed in most Fast Path implementations.

7. IANA Considerations

There are no actions required for IANA defined in this document.

8. Security Considerations

Security issues with IPv6 Hop-by-Hop options are well known and have been documented in several places, including [RFC6398], [RFC6192], and [I-D.ietf-v6ops-ipv6-ehs-packet-drops]. The main issue, as noted in Section 4, is that any mechanism that can be used to force packets into the router's Slow Path can be exploited as a denial of service attack on a transit router by saturating the resources need for router management protocols (e.g., routing protocols, network management protocols, etc.) that may cause the router to fail. Due to this it's common for transit routers to drop packets with Hop-by-Hop options headers.

While Hop-by-Hop options are not required to be processed in the Slow Path, the Router Alert options is designed to do just that.

This document changes the way Hop-by-Hop options are processed in several ways that significantly reduces the attack surface. These changes include:

- * All Hop-by-Hop options (with one exception) must be processed in the Fast Path. Only one HBH Option MUST be processed and additional HBH Options MAY be processed based on local configuration.
- * Only the Router Alert option can be processed in the Slow Path, and the router must be configured to do so.
- * Added criteria to allow control over how Router Alert options are processed and that a node configured to support these options must protect itself from attacks using the Router Alert.
- * Limited the default number of Hop-by-Hop options that that can be in a packet to a single Hop-by-Hop option.
- * Additional Hop-by-Hop options MAY be included, based on local configuration. Although nodes only process these additional Hop-by-Hop Options if configured to do so.
- * Added restrictions to any future new Hop-by-Hop options that limit their size and computational requirements.

The authors believe that these changes significantly reduces the security issues relating to IPv6 Hop-by-Hop options and will enable them to be used safely in the Internet.

9. Acknowledgments

Helpful comments were received from Brian Carpenter, Ron Bonica, Ole Troan, Mark Heard, Tom Herbert, [your name here], and other members of the 6MAN working group.

10. Change log [RFC Editor: Please remove]

draft-hinden-6man-hbh-processing-01, 2021-June-2:

- * Expanded terminology section to include Forwarding Plane and Control Plane.
- * Changed draft that only one HBH Option MUST be processed and additional HBH Options MAY be processed based on local configuration.
- * Clarified that all HBH options (with one exception) must be processed on the Fast Path.
- * Kept the Router Alert options as the single exception for Slow Path processing.
- * Rewrote and expanded section on New Hop-by-Hop Options.
- * Removed requirement for HBH Option size and alignment.
- * Removed sections evaluating currently defined HBH Options.
- * Added content to the Security Considerations section.
- * Added people to the acknowledgements section.
- * Numerous editorial changes

draft-hinden-6man-hbh-processing-00, 2020-Nov-29:

- * Initial draft.

11. Normative References

- [IANA-HBH] "Destination Options and Hop-by-Hop Options",
<<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200,

DOI 10.17487/RFC8200, July 2017,
<<https://www.rfc-editor.org/info/rfc8200>>.

12. Informative References

- [Hendriks] Hendriks, L., Velan, P., Schmidt, R.O., Boer, P., and A. Aiko, "Threats and Surprises behind IPv6 Extension Headers", August 2017,
<http://dl.ifip.org/db/conf/tma/tma2017/tma2017_paper22.pdf>.
- [I-D.ietf-v6ops-ipv6-ehs-packet-drops] Gont, F., Hilliard, N., Doering, G., Kumari, W., Huston, G., and W. (. Liu, "Operational Implications of IPv6 Packets with Extension Headers", Work in Progress, Internet-Draft, draft-ietf-v6ops-ipv6-ehs-packet-drops-06, 8 April 2021, <<https://tools.ietf.org/html/draft-ietf-v6ops-ipv6-ehs-packet-drops-06>>.
- [IANA-RA] "IPv6 Router Alert Option Values", <<https://www.iana.org/assignments/ipv6-routeralert-values/ipv6-routeralert-values>>.
- [RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, DOI 10.17487/RFC2711, October 1999, <<https://www.rfc-editor.org/info/rfc2711>>.
- [RFC6192] Dugal, D., Pignataro, C., and R. Dunn, "Protecting the Router Control Plane", RFC 6192, DOI 10.17487/RFC6192, March 2011, <<https://www.rfc-editor.org/info/rfc6192>>.
- [RFC6398] Le Faucheur, F., Ed., "IP Router Alert Considerations and Usage", BCP 168, RFC 6398, DOI 10.17487/RFC6398, October 2011, <<https://www.rfc-editor.org/info/rfc6398>>.

Authors' Addresses

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

Email: bob.hinden@gmail.com

Godred Fairhurst
University of Aberdeen

School of Engineering, Fraser Noble Building
Aberdeen
AB24 3UE
United Kingdom

Email: gorry@erg.abdn.ac.uk

6MAN Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 30, 2022

G. Fioccola
T. Zhou
Huawei
M. Cociglio
Telecom Italia
F. Qin
China Mobile
R. Pang
China Unicom
April 28, 2022

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

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. Terminology	3
1.2. Requirements Language	3
2. Alternate Marking application to IPv6	3
2.1. Controlled Domain	5
2.1.1. Alternate Marking Measurement Domain	6
3. Definition of the AltMark Option	7
3.1. Data Fields Format	7
4. Use of the AltMark Option	8
5. Alternate Marking Method Operation	10
5.1. Packet Loss Measurement	10
5.2. Packet Delay Measurement	12
5.3. Flow Monitoring Identification	13
5.4. Multipoint and Clustered Alternate Marking	15
5.5. Data Collection and Calculation	16
6. Security Considerations	16
7. IANA Considerations	20
8. Acknowledgements	20
9. References	20
9.1. Normative References	20
9.2. Informative References	21
Authors' Addresses	22

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:

- o 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.
- o 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.
- o 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

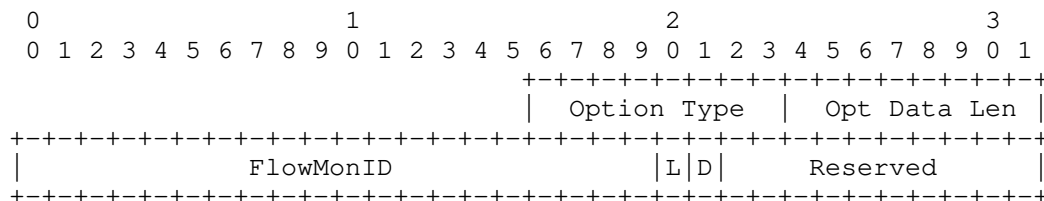
boundaries, for example a simple security check can be to insert the Alternate Marking data if and only if the destination is within the controlled domain.

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).



where:

- o 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.
- o Opt Data Len: 4. It is the length of the Option Data Fields of this Option in bytes.

- o 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.
- o L: Loss flag for Packet Loss Measurement as described in Section 5.1;
- o D: Delay flag for Single Packet Delay Measurement as described in Section 5.2;
- o Reserved: is reserved for future use. These bits MUST be set to zero on transmission and ignored on receipt.

4. Use of the AltMark Option

The AltMark Option is the best way to implement the Alternate Marking method and 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:

- o Destination Option not preceding a Routing Header => measurement only by node in Destination Address.
- o Hop-by-Hop Option => every router on the path with feature enabled.
- o 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 $\pm B/2$ 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.

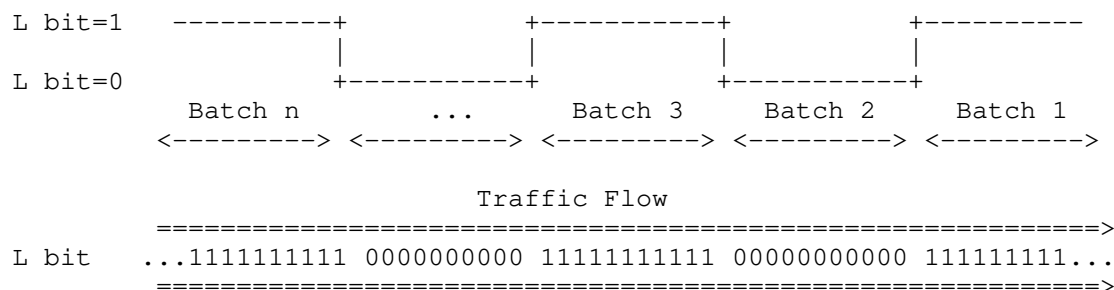


Figure 1: Packet Loss Measurement and Single-Marking Methodology using L bit

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.

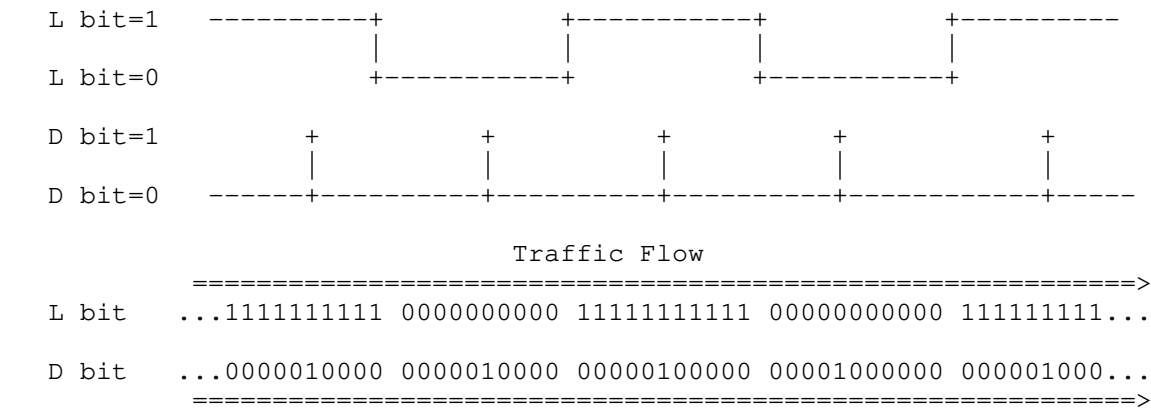


Figure 2: Double-Marking Methodology using L bit and D bit

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:

- o 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.
- o 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/>).

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

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]
Fioccola, G., Cociglio, M., Mirsky, G., Mizrahi, T., and T. Zhou, "Alternate-Marking Method", draft-ietf-ippm-rfc8321bis-01 (work in progress), April 2022.
- [I-D.ietf-ippm-rfc8889bis]
Fioccola, G., Cociglio, M., Sapio, A., Sisto, R., and T. Zhou, "Multipoint Alternate-Marking Clustered Method", draft-ietf-ippm-rfc8889bis-01 (work in progress), April 2022.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

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

9.2. Informative References

- [I-D.chen-pce-pcep-ifit]
Yuan, H., Zhou, T., Li, W., Fioccola, G., and Y. Wang, "Path Computation Element Communication Protocol (PCEP) Extensions to Enable IFIT", draft-chen-pce-pcep-ifit-06 (work in progress), February 2022.
- [I-D.fz-spring-srv6-alt-mark]
Fioccola, G., Zhou, T., and M. Cociglio, "Segment Routing Header encapsulation for Alternate Marking Method", draft-fz-spring-srv6-alt-mark-02 (work in progress), February 2022.
- [I-D.hinden-6man-hbh-processing]
Hinden, R. M. and G. Fairhurst, "IPv6 Hop-by-Hop Options Processing Procedures", draft-hinden-6man-hbh-processing-01 (work in progress), June 2021.
- [I-D.ietf-idr-sr-policy-ifit]
Qin, F., Yuan, H., Zhou, T., Fioccola, G., and Y. Wang, "BGP SR Policy Extensions to Enable IFIT", draft-ietf-idr-sr-policy-ifit-03 (work in progress), January 2022.
- [I-D.peng-v6ops-hbh]
Peng, S., Li, Z., Xie, C., Qin, Z., and G. Mishra, "Processing of the Hop-by-Hop Options Header", draft-peng-v6ops-hbh-06 (work in progress), August 2021.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<https://www.rfc-editor.org/info/rfc6437>>.

- [RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <<https://www.rfc-editor.org/info/rfc6438>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<https://www.rfc-editor.org/info/rfc7045>>.
- [RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <<https://www.rfc-editor.org/info/rfc7384>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.
- [RFC8799] Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", RFC 8799, DOI 10.17487/RFC8799, July 2020, <<https://www.rfc-editor.org/info/rfc8799>>.
- [RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <<https://www.rfc-editor.org/info/rfc8915>>.

Authors' Addresses

Giuseppe Fioccola
Huawei
Riesstrasse, 25
Munich 80992
Germany

Email: giuseppe.fioccola@huawei.com

Tianran Zhou
Huawei
156 Beiqing Rd.
Beijing 100095
China

Email: zhoutianran@huawei.com

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

Email: mauro.cociglio@telecomitalia.it

Fengwei Qin
China Mobile
32 Xuanwumenxi Ave.
Beijing 100032
China

Email: qinfengwei@chinamobile.com

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

Email: pangran@chinaunicom.cn

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 11 November 2022

R. Hinden
Check Point Software
G. Fairhurst
University of Aberdeen
10 May 2022

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.

Table of Contents

1. Introduction	2
1.1. Example Operation	3
1.2. Use of the IPv6 Hop-by-Hop Options Header	4
2. Motivation and Problem Solved	5
3. Requirements Language	6
4. Applicability Statements	6
5. IPv6 Minimum Path MTU Hop-by-Hop Option	6
6. Router, Host, and Transport Layer Behaviors	8
6.1. Router Behavior	8
6.2. Host Operating System Behavior	8
6.3. Transport Layer Behavior	9
6.3.1. Including the Option in an Outgoing Packet	10
6.3.2. Validation of the Packet that includes the Option	12
6.3.3. Receiving the Option	12
6.3.4. Using the Rtn-PMTU Field	13
6.3.5. Detecting Path Changes	14
6.3.6. Detection of Dropping Packets that include the Option	14
7. IANA Considerations	14
8. Security Considerations	14
8.1. Router Option Processing	15
8.2. Network Layer Host Processing	15
8.3. Validating use of the Option Data	16
8.4. Direct use of the Rtn-PMTU Value	16
8.5. Using the Rtn-PMTU Value as a Hint for Probing	17
8.6. Impact of Middleboxes	17
9. Experiment Goals	17
10. Implementation Status	18
11. Acknowledgments	18
12. Change log [RFC Editor: Please remove]	18
13. References	21
13.1. Normative References	21
13.2. Informative References	22
Appendix A. Examples of Usage	24
Authors' Addresses	26

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.

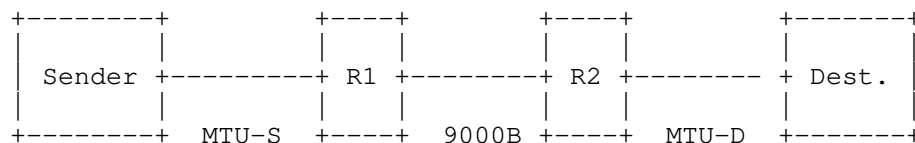


Figure 1

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.

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

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 McCloghrie. 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	Option Data Len	Option Data		
BBCTTTTT	00000100	Min-PMTU	Rtn-PMTU	R

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

BB 00 Skip over this option and continue processing.

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

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

Length: 4 The size of the 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 DPLPMTUD [RFC8899] (with the important difference that a PTB message can only seek to lower the PMTU, whereas this option could trigger a probe packet to seek to increase the PMTU.)

Section 5.2 of [RFC8201] provides guidance on the caching of PMTU information and also the relation to IPv6 flow labels. Implementations should consider the impact of Equal Cost Multipath (ECMP) [RFC6438]. Specifically, whether a PMTU ought 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.
 - Zaheduzzaman Sarker's Review: Fixed some editorial issues, and updated SHOULD language.
- * 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.

draft-ietf-6man-mtu-option-09, 2021-September-23

- * Clarifications and editorial changes based on review by Michael Dougherty.

draft-ietf-6man-mtu-option-08, 2021-September-7

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

draft-ietf-6man-mtu-option-07, 2021-August-31

- * 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.

draft-ietf-6man-mtu-option-06, 2021-August-7

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

draft-ietf-6man-mtu-option-05, 2021-April-28

- * Editorial changes.

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

- * Fixes for typos.

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

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

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

- * Editorial changes to make text and terminology more consistent.

- * Added reference to DPLPMTUD.

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

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

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

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

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

- * Changed option format to also include the Returned PMTU value and Return flag and made related text changes in Section 6.2 to describe this behavior.
- * 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

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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.

13.2. Informative References

- [I-D.ietf-taps-arch] Pauly, T., Trammell, B., Brunstrom, A., Fairhurst, G., and C. Perkins, "An Architecture for Transport Services", Work in Progress, Internet-Draft, draft-ietf-taps-arch-12, 3 January 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-taps-arch-12>>.
- [RFC1063] Mogul, J., Kent, C., Partridge, C., and K. McCloghrie, "IP MTU discovery options", RFC 1063, DOI 10.17487/RFC1063, July 1988, <<https://www.rfc-editor.org/info/rfc1063>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.

- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <<https://www.rfc-editor.org/info/rfc4821>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <<https://www.rfc-editor.org/info/rfc6438>>.
- [RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <<https://www.rfc-editor.org/info/rfc7637>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8899] Fairhurst, G., Jones, T., Tüxen, M., Rüngeler, I., and T. Völker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <<https://www.rfc-editor.org/info/rfc8899>>.
- [RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <<https://www.rfc-editor.org/info/rfc8900>>.
- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/info/rfc9000>>.

- [RFC9099] Vyncke, É., Chittimaneni, K., Kaeo, M., and E. Rey,
"Operational Security Considerations for IPv6 Networks",
RFC 9099, DOI 10.17487/RFC9099, August 2021,
<<https://www.rfc-editor.org/info/rfc9099>>.
- [VPP] "VPP/What is VPP?",
<https://wiki.fd.io/view/VPP/What_is_VPP%3F>.
- [VPP_SRC] "VPP Source", <<https://gerrit.fd.io/r/c/vpp/+/21948>>.
- [WIRESHARK] "Wireshark Network Protocol Analyzer",
<<https://www.wireshark.org>>.

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.

```

----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

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') ----->

```

Figure 5

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 -----X
----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) -----X
<--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') ----->

```

Figure 6

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

Email: bob.hinden@gmail.com

Godred Fairhurst
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen
AB24 3UE
United Kingdom
Email: gorrry@erg.abdn.ac.uk

IPv6 Maintenance (6man) Working Group
Internet-Draft
Updates: 4861, 4862 (if approved)
Intended status: Standards Track
Expires: July 23, 2021

F. Gont
SI6 Networks
J. Zorz
6connect
R. Patterson
Sky UK
January 19, 2021

Improving the Robustness of Stateless Address Autoconfiguration (SLAAC)
to Flash Renumbering Events
draft-ietf-6man-slaac-renum-02

Abstract

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

Status of This Memo

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 July 23, 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	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	16
A.1. On a Possible Reaction to ICMPv6 Error Messages	16
A.2. On a Possible Improvement to Source Address Selection . .	17
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 [I-D.ietf-v6ops-slaac-renum]. In such scenarios, hosts on the local network will continue using stale prefixes for an unacceptably long period of time, thus resulting in connectivity problems. [I-D.ietf-v6ops-slaac-renum] discusses this problem in detail.

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

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

- o Avoiding the use of stale addresses for new communication instances
- o Performing "garbage collection" for the stale prefixes (and related network configuration information)

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

2. Terminology

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

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

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

3. SLAAC reaction to Flash-renumbering Events

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

3.1. Renumbering without Explicit Signaling

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

- o Preferred Lifetime (AdvPreferredLifetime): 604800 seconds (7 days)
- o Valid Lifetime (AdvValidLifetime): 2592000 seconds (30 days)

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

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

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

3.2. Renumbering with Explicit Signaling

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

- o The amount of time during which the router should continue trying to deprecate the stale network configuration information
- o The ability of SLAAC hosts to phase out stale configuration in a timelier manner.

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

NOTE:

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

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

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

Section 4.2 updates [RFC4862] such that this restriction 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.

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

4.1. More Appropriate Lifetime Values

4.1.1. Router Configuration Variables

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

```
AdvPreferredLifetime: max(AdvDefaultLifetime, 3 *  
MaxRtrAdvInterval)
```

```
AdvValidLifetime: 2 * AdvPreferredLifetime
```

where:

AdvPreferredLifetime:

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

AdvValidLifetime:

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

AdvDefaultLifetime:

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

max():

A function that computes the maximum of its arguments.

NOTE:

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

RATIONALE:

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

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

- * In the context of [RFC8028], where it is clear that the use of addresses configured for a given prefix is tied to the next-hop router that advertised the prefix, the "Preferred Lifetime" of a PIO should not be larger than the "Router Lifetime" of Router Advertisement messages. Some leeway should be provided for the "Valid Lifetime" of PIOs, to cope with transient network problems. As a result, this document updates [RFC4861] such that the default Valid Lifetime (AdvValidLifetime) and the default Preferred Lifetime (AdvPreferredLifetime) of PIOs are specified as a function of the "Router Lifetime" (AdvDefaultLifetime) of Router Advertisement messages. In the absence of RAs that refresh information, addresses configured for previously-advertised prefixes become deprecated in a timelier manner, and thus Rule 3 of [RFC6724] will cause other configured addresses (if available) to be preferred.
- * The expression above computes the maximum between AdvDefaultLifetime and " $3 * \text{MaxRtrAdvInterval}$ " (the default value of AdvDefaultLifetime, as per [RFC4861]) to cope with the case where an operator might simply want to disable one local router for maintenance, without disabling the use of the corresponding prefixes on the local network (e.g., on a multi-router network). [RFC4862] implementations would otherwise deprecate the corresponding prefixes. Similarly, [RFC8028] implementations would likely behave in the same way.

4.2. Honor Small PIO Valid Lifetimes

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

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

RATIONALE:

- * This change allows hosts to react to the information provided by a router that has positive knowledge that a prefix has become invalid.
- * The behavior described in RFC4862 had been incorporated during the revision of the original IPv6 Stateless Address

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

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

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

4.3. Interface Initialization

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

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

with:

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

RATIONALE:

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

NOTE:

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

4.4. Conveying Information in Router Advertisement (RA) Messages

[TBD]

4.5. Recovery from Stale Configuration Information without Explicit Signaling

[TBD]

5. IANA Considerations

This document has no actions for IANA.

6. Implementation Status

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

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

6.1. More Appropriate Lifetime Values

6.1.1. Router Configuration Variables

6.1.1.1. rad(8)

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

6.1.1.2. radvd(8)

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

- o Preferred Lifetime: 14400 seconds (4 hours)
- o Valid Lifetime: 86400 seconds (1 day)

This is not following the specific recommendation in this document, 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:

- o Rather than recording prefixes on stable storage (as recommended in [I-D.ietf-v6ops-cpe-slaac-renum]), this implementation checks the source address of IPv6 packets, and assumes that usage of any address that does not correspond to a prefix currently-advertised

by the Customer Edge Router is the result of stale network configuration information. Hence, upon receipt of a packet that employs a source address that does not correspond to a currently-advertised prefix, this implementation will start advertising the corresponding prefix with small lifetimes, with the intent of deprecating it.

- o 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 [I-D.ietf-v6ops-slaac-renum], and eventually to this document.

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

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC7772] Yourtchenko, A. and L. Colitti, "Reducing Energy Consumption of Router Advertisements", BCP 202, RFC 7772, DOI 10.17487/RFC7772, February 2016, <<https://www.rfc-editor.org/info/rfc7772>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<https://www.rfc-editor.org/info/rfc8028>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

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

9.2. Informative References

- [dhcpcd] Marples, R., "dhcpcd - a DHCP client", <<https://roy.marples.name/projects/dhcpcd/>>.
- [FRITZ] Gont, F., "Quiz: Weird IPv6 Traffic on the Local Network (updated with solution)", SI6 Networks Blog, February 2016, <<https://www.si6networks.com/2016/02/16/quiz-weird-ipv6-traffic-on-the-local-network-updated-with-solution/>>.
- [I-D.ietf-v6ops-cpe-slaac-renum] Gont, F., Zorz, J., Patterson, R., and B. Volz, "Improving the Reaction of Customer Edge Routers to Renumbering Events", draft-ietf-v6ops-cpe-slaac-renum-06 (work in progress), December 2020.
- [I-D.ietf-v6ops-slaac-renum] Gont, F., Zorz, J., and R. Patterson, "Reaction of Stateless Address Autoconfiguration (SLAAC) to Flash-Renumbering Events", draft-ietf-v6ops-slaac-renum-05 (work in progress), November 2020.
- [IPNG-minutes] IETF, "IPNG working group (ipngwg) Meeting Minutes", Proceedings of the thirty-eighth Internet Engineering Task Force , April 1997, <<https://www.ietf.org/proceedings/38/97apr-final/xrtftr47.htm>>.
- [NetworkManager] NetworkManager, "NetworkManager web site", <<https://wiki.gnome.org/Projects/NetworkManager>>.
- [rad] Obser, F., "OpenBSD Router Advertisement Daemon - rad(8)", <<https://cvsweb.openbsd.org/src/usr.sbin/rad/>>.
- [radvd] Hawkins, R. and R. Johnson, "Linux IPv6 Router Advertisement Daemon (radvd)", <<http://www.litech.org/radvd/>>.

- [radvd.conf] Hawkins, R. and R. Johnson, "radvd.conf - configuration file of the router advertisement daemon", <<https://github.com/reubenhwk/radvd/blob/master/radvd.conf.5.man>>.
- [RFC1971] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 1971, DOI 10.17487/RFC1971, August 1996, <<https://www.rfc-editor.org/info/rfc1971>>.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<https://www.rfc-editor.org/info/rfc2827>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, DOI 10.17487/RFC5927, July 2010, <<https://www.rfc-editor.org/info/rfc5927>>.
- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <<https://www.rfc-editor.org/info/rfc6105>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC7113] Gont, F., "Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)", RFC 7113, DOI 10.17487/RFC7113, February 2014, <<https://www.rfc-editor.org/info/rfc7113>>.
- [slaacd] Obser, F., "OpenBSD SLAAC Daemon - slaacd(8)", <<https://cvsweb.openbsd.org/src/usr.sbin/slaacd/>>.
- [systemd] systemd, "systemd web site", <<https://systemd.io/>>.

Appendix A. Analysis of Some Suggested Workarounds

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

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

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

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

The aforementioned proposal has a number of drawbacks and limitations:

- o It assumes that the CPE routers enforce ingress/egress filtering [RFC2827]. While this is desirable behaviour, it cannot be relied upon.
- o It assumes that if the CPE enforces ingress/egress filtering, the CPE will signal the packet drops to the sending node with ICMPv6 Type 1 (Destination Unreachable) Code 5 (Source address failed ingress/egress policy) error messages. While this may be desirable, [RFC2827] does not suggest signaling the packet drops with ICMPv6 error messages, let alone the use of specific error messages (such as Type 1 Code 5) as suggested.
- o ICMPv6 Type 1 Code 5 could be interpreted as the employed address being stale, but also as a selected route being inappropriate/suboptimal. If the later, deprecating addresses or invalidating addresses upon receipt of these error messages would be inappropriate.
- o Reacting to these error messages would create a new attack vector that could be exploited from remote networks. This is of particular concern since ICMP-based attacks do not even require that the Source Address of the attack packets be spoofed [RFC5927].

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

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

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

- o The node is assumed to maintain a timer/counter that is updated at least once per second. For example, the time(2) function from unix-like systems could be employed for this purpose.
- o The local information associated with each prefix advertised via RAs on the local network is augmented with a "LastAdvertised" timestamp value. Whenever an RA with a PIO with the "A" bit set for such prefix is received, the "LastAdvertised" timestamp is updated with the current value of the timer/counter.
- o [RFC6724] is updated such that this rule is incorporated:

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

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

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

- o In scenarios where multiple prefixes are being advertised on the same LAN segment, the new SAS rule is *guaranteed* to result in non-deterministic behaviour, with hosts frequently changing the default source address. This is certainly not desirable from a troubleshooting perspective.
- o Since the rule must be incorporated before "Rule 8: Use longest matching prefix" from [RFC6724], it may lead to suboptimal paths.
- o This new rule may help to improve the selection of a source address, but it does not help with the housekeeping (garbage collection) of configured information:

- * If the stale prefix is re-used in another network, nodes employing stale addresses and routes for this prefix will be unable to communicate with the new "owner" of the prefix, since the stale prefix will most likely be considered "on-link".
- * Given that the currently recommended default value for the "Valid Lifetime" of PIOs is 2592000 seconds (30 days), it would take too long for hosts to remove the configured addresses and routes for the stale prefix. While the proposed update in Section 4.1 of this document would mitigate this problem, the lifetimes advertised by the local SLAAC router are not under the control of hosts.

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

Authors' Addresses

Fernando Gont
SI6 Networks
Seguro 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

6MAN Working Group
Internet-Draft
Updates: RFC2464, RFC4291, RFC4861, RFC4862,
RFC7136, RFC8273 (if approved)
Intended status: Standards Track
Expires: 1 November 2022

G. Mishra
Verizon Inc.
A. Petrescu
CEA, LIST
N. Kottapalli
Benu Networks
D. Mudric
Ciena
D. Shytyi
SFR
30 April 2022

SLAAC with prefixes of arbitrary length in PIO (Variable SLAAC)
draft-mishra-6man-variable-slaac-06

Abstract

This draft proposes the use of arbitrary length prefixes in PIO for SLAAC. A prefix of length 63 in PIO, for example, would be permitted to form an address whose interface identifier length is 65, which allows several benefits. A prefix of length 65 would be allowed too, but it SHOULD NOT be used on a large scale, like at a large ISP; this is to avoid a race to the bottom.

The implementation uses a parameter in the Host; this option is off by default. In that case, the Host respects the 64bit boundary. When the parameter is set to on the Host accepts prefixes of lengths different than 64 and forms 128bit addresses.

In the past, various IPv6 addressing models have been proposed based on a subnet hierarchy embedding a 64-bit prefix. The last remnant of IPv6 classful addressing is a inflexible interface identifier boundary at /64. This document proposes flexibility to the fixed position of that boundary for interface addressing.

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 1 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.

Table of Contents

1. Terminology	3
2. Introduction	3
3. The History behind the 64 bit fixed boundary	4
4. Identifier and Subnet Length Statements	8
5. Recommendations for implementation of variable SLAAC	8
6. Recommended use cases where 64 bit prefix should be utilized	9
7. Reasons for longer than 64 bit prefix length	13
7.1. Insufficient Address Space Delegated	13
7.2. Hierarchical Addressing	14
7.3. Audit Requirement	14
7.4. Concerns over ND Cache Exhaustion	15
7.5. Longer prefixes lengths used for embedding information	15
8. Greater than 64 bit prefix usage by ISPs is strictly prohibited	15
9. Comparison of Static, SLAAC, DHCPv6 and Variable SLAAC	15
10. Variable SLAAC Use Cases	18
10.1. Permission-less Extension of the Network	18
10.2. Private Networks	18
10.3. Mobile IPv6	19
10.4. Home and SOHO	19
10.5. 3GPP V2I and V2V networking	19
10.6. 6lo	20
10.7. Large ISP's backbone POP	20
11. Variable SLAAC implementation	20

12. Applicability Statements	21
13. Router and Operational Considerations	21
14. Host Behavior Considerations	21
15. Security Considerations	22
16. IANA Considerations	22
17. Contributors	22
18. Acknowledgements	22
19. References	22
19.1. Normative References	22
19.2. Informative References	30
Appendix A. ChangeLog	30
Authors' Addresses	31

1. Terminology

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. Introduction

From the beginning, the IPv6 addressing plan was based on a 128 bit address format made up of 8 hexets which were broken down into a 64 bit four hexet prefix and 64 bit four hexet interface identifier. For example, the address 2001:db8:3:4::1 has the first 4 hexets forming the /64 prefix 2001:db8:3:4::/64, whereas the last four hexets form an interface identifier abbreviated as ::1 (a 'hexet' is a group of max 4 hex digits between two columns, e.g. "2001" and "db8" are each a hexet). A comprehensive analysis of the 64-bit boundary is provided in [RFC7421]. The history of IPv6 Classful models proposed, and the last remnant of IPv6 Classful addressing rigid network interface identifier boundary at /64 is discussed in detail as well as the removal of the fixed position of the boundary for interface addressing in draft [I-D.bourbaki-6man-classless-ipv6].

This document discusses the reasons why the interface identifier has been fixed at 64 bits, and the problems that can be addressed by changing the GUA interface identifier from fixed 64 bit size to a variable interface identifier. This change would be consistent with static and DHCPv6 stateful IPv6 address assignment, as well as the proposed solution would ensure maintaining backwards compatibility for existing implementations. This document tries to achieve clearing the confusion related to prefix length, and provide consistency of variable length prefix across the three IPv6 addressing strategies deployed, static, DHCPv6 and Stateless Address Autoconfiguration (SLAAC), and finally update all RFCs with the new variable SLAAC standard. The 64 bit fixed boundary problem statement is defined in draft [I-D.mishra-v6ops-variable-slaac-problem-stmt].

Over the years one of the merits of increasing the prefix length, and reducing the size of the interface identifier has been incorrectly stated as the possibility of IPv6 address space exhaustion could be circumvented, or that a 64 bit interface identifier is a wasteful use of address space.

3. The History behind the 64 bit fixed boundary

The fixed length of an Interface Identifier has roots in other early non-IP networks such as IPX of Novell and another from Apple.

Over the course of the history of the IPv6 protocol, several addressing models have been proposed to break up the prefix into a hierarchical format. One of the first attempts was [RFC2450] which was based on a 13 bit Level Aggregation (TLA), 24 bit Next-Level Aggregation (NLA), 16 bit Site Level aggregator Identifiers. The current IPv6 addressing architecture for global unicast addressing uses [RFC3587] for global unicast address currently being delegated by IANA 2000::/3 prefix. With the recommendation in [RFC3177] which called for a default end site assignment of a /48 which was adopted by the Regional Internet Registry was revised with [RFC6177] to a smaller block size of /56 prefix to end sites to avoid risk of premature address depletion. The current IPv6 addressing architecture [RFC3587] for global unicast addressing was now based on an IPv6 hierarchical format which now consists of a 45 bit global routing prefix, 16 bit subnet ID followed by 64 bit interface identifier. In the earlier deployments of IPv6 due to the stringent guidelines of [RFC4291] which stated that for all unicast addresses, except those that start with the binary value 000, Interface IDs are required to be 64 bits long and to be constructed in Modified EUI-64 format. Referencing IPv6 Addressing architecture [RFC3513] section 2.5.5 depicts examples of global unicast addresses that start with binary 000 are IPv6 addresses with embedded IPv4 addresses and IPv6 address containing encoded NSAP addresses [RFC4548] described in

[RFC6052]. An example use case would be for NAT64 [RFC6146] as well as many other use cases that exist with transition technology tunneling using IPv4 IPv6 translators.

The general format for IPv6 global unicast addresses is as follows:

	n bits		m bits		128-n-m bits	
+	-----	+	-----	+	-----	+
	global routing prefix		subnet ID		interface ID	
+	-----	+	-----	+	-----	+

Figure 1: Format of the IPv6 global unicast addresses

Even though [RFC4291] states that all global unicast addresses except those that start with binary value 000, which use ipv4 ipv6 translators [RFC6052], that static and DHCPv6 violates [RFC4291] as variable length masking to 128 is supported, where SLAAC variable length masking remains forbidden. IPv6 packets over LAN based technologies such as ethernet must use 64 bit interface identifier per [RFC2464]. Nothing is mentioned regarding wireless based technologies such as MIP6, V2V or 6LoWPAN, with regards to interface identifier length stringent requirement for 64 bit prefix length. Stateful Address Autoconfiguration [RFC4862] states that the sum total of the prefix length and interface identifier should equal 128 bits, but does not state that the interface identifier should be 64 bits. Note that [RFC4861] states that the PIO (Prefix information Options), that the A-bit Autonomous address-configuration flag when set indicates that the prefix can be used for (SLAAC) stateless address autoconfiguration, and [RFC4862] states to silently ignore the PIO options if the A flag is not set in the Router Advertisement. If the A flag is not set then /64 is only a recommendation which applies to DHCPv6 and static.

During the early deployments of IPv6, /64 was a 'de facto' standard prefix length for deployment to all router interfaces including point-to-point and loopbacks. In early deployments of IPv6, due to the complexity and overall learning curve, and change going from IPv4 to IPv6, the keep it simple approach of /64 everywhere was the general rule of thumb for deployment. After decades of deployment, operators started to dig further into how IPv4 started out as classful with classful routing protocols such as RIP or IGRP. Later as Classless inter-domain routing with BGP became mainstream with larger enterprises and service providers, operators started looking at IPv6 and variable length masking. Operators now started experimenting trying to subnet at nibble boundaries to start and became brave enough to tackle subnetting on a bit boundary. As

variable length subnet masking became more mainstream with IPv6, operators started to use /126 mask on point-to-point links. Around that time [RFC3627] came out which talked about the harmful effects of /127 and that it was forbidden due to operational impacts. Harmful impacts of /127 were due to subnet-router anycast being in conflict with [RFC2526] /121. Later was found the benefits of /127 avoided the ping-pong effect and the subnet-router anycast conflict could be avoided by disabling Duplicate address detection thus the status of use of /127 on point-to-point links was updated by [RFC6164]. As the evolution of IPv6 continued, questions would come as to why the interface identifier is so large at 64 bits, as 64 bits equates to 18,446,744,073,709,551,616 IPv6 addresses, which is more than anyone could ever imagine on a single flat subnet far into the distant future. The main reason for the larger 64 bit interface identifier is for privacy when connected directly to the internet, or on an unsecure public hotspot or location so your device is not traceable.

From the beginning of IPv6 deployments most enterprises went with SLAAC, but as DHCPv6 matured, enterprises migrated to DHCPv6, and network infrastructure remained configured manually using static configurations. Since so many RFC's mention the SLAAC 64 bit boundary requirement and confusion related to this topic, in fact prevented operators proliferation of even attempting to use longer prefixes on host subnets with static or DHCPv6 stateful. Most IPv6 implementations even to this day do not use longer than 64 bit prefixes, and still maintain the 64 bit boundary for host subnet, for both DHCPv6 and static, even though technically feasible, due to fear of interoperability issues that may arise. With this new evolution of IPv6 addressing architecture with variable SLAAC, we can bring back SLAAC to the mainstream for all IPv6 deployments. This will also allow operators to now comfortably deploy both DHCPv6 and static with greater than 64 bit prefix length to host subnets, without fear of interoperability problems.

Today we have three methods of IPv6 address deployment, SLAAC, DHCPv6 and static. DHCPv6 does not provide an adequate IPv6 addressing solution as described in detail in the DHCPv6, Static, and SLAAC comparison section. As user subnets flatten out further, as the IPv4 under pinning is eliminated, removing the shackles on IPv6, the subnets will get much flatter. As the subnets flatten out in large Enterprise networks where you have 100's of Dual Stack subnets migrate to a single "IPv6-ONLY" subnet, the overhead DHCPv6 Normal mode messaging becomes exacerbated. The problem with DHCPv6 is that once the "M" managed bit is set to "1", all hosts on the subnet cache the M bit and change to DHCPv6 stateful mode. Higher probability of rouge devices such as printers or other appliances misbehaving with IPv6 enabled by default, now in DHCPv6 mode, spewing of millions of

DHCPv6 messages that can now impact the router control plane processing of packets. This can be alleviated with special custom Control Plane policer policy, however now adds complexity and administrative overhead to DHCPv6 deployments. Enterprises and Service Providers require a viable IPv6 deployment solution that can accommodate the shortfalls of both static and DHCPv6 addressing. Static addressing due to administrative overhead of manual assignment does not provide a viable solution for even moderately sized networks.

An arbitrary length prefix solves problems described in detail in section 7 and are being highlighted here as well as a key part of the problem statement to be addressed. A site may not be able to delegate sufficient address space from a /64 prefix to all of its internal subnets. In this case a site may be partially operational as it is unable to number all of its subnets. An alternative would be to be able to use prefixes longer than /64 to allow multiple subnets for example /80 for numbering subnets with a mixture of hosts that are static or DHCPv6 without worry of interoperability issues. Some operators would like the ability to have a hierarchical addressing structure and may require more than 16 bits given with a /48 allocation. In such instances longer prefix lengths would allow for additional levels of aggregation as required. It is common for some operators to have security audit requirements where they wish to know all active hosts on a /64 subnet. As /64 subnets can contain an enormous number of hosts and thus cannot be scanned as can IPv4 subnets. Operators have argued that one method to be able to scan for active hosts would be by reducing the size of the subnet. Neighbor discovery cache exhaustion when an attacker sends a large number of messages in rapid succession to hosts filling the routers ND cache is another problem with fixed length /64 size SLAAC subnets. Neighbor Discovery cache exhaustion issues are relatively common on IXP (Internet Exchange Points) where a very large number of Internet Service Providers are full mesh peering to exchange routing updates. As the number of hosts on a SLAAC subnet can be 2^{64} , a much smaller subnet size can drastically reduce the Neighbor Discovery cache exhaustion issues.

The goal of this document is to fix the problems related to stateless address autoconfiguration (SLAAC), current obscurities of the 64 bit prefix boundary, issues that exist today with current IPv6 addressing using manual and DHCPv6, and how variable SLAAC can now be used to fill the gaps with static and DHCPv6, and also update all standards specifications to reflect the new variable SLAAC standard making the prefix lengths variable.

4. Identifier and Subnet Length Statements

IPv6 router interfaces and hosts configured to use Stateful Address Autoconfiguration (SLAAC) will now support variable mask up to 128 bits consistent with static and DHCPv6. This change will allow variable SLAAC to be used on any infrastructure link from point-to-point /127 to infrastructure shared subnets from /65 to /127. All routers support routing of variable length IPv6 prefix lengths called variable length subnet masks (VLSM) up to 128 bits in length, so this variable stateless address autoconfiguration change will be in line with all interior gateway routing protocols and exterior gateway routing protocols. This change is for both Global Unicast address [RFC3587] and Unique Local Address [RFC4193]. There will be no change to the IPv6 link local address interface identifier which will remain 64 bits for link local fe80::/10 router or host interface fe80::/64 [RFC4291].

The term "Variable SLAAC" as defined in this document states that the length of the prefix can now be greater than 64 bits up to 127 bits with a corresponding shorter interface identifier. The interface identifier will range from 64 bits to 1 bit in length. The length of the prefix can now be less than 64 bits to 3 bits in length with a corresponding longer interface identifier and can now be greater than 64 bits to 125 bits in length.

The "race to the bottom problem" - is the problem where allowing prefixes longer than 64 to be used in SLAAC will lead to 65, 66 and so on, up to 127 and 128 allocations. At that point the bottom would be reached and thus impossible to extend the network further.

One version of the "address waste" problem is: SLAAC is used in a subnet where 2^{64} addresses are possible. But there are no link layers that allow as many addresses to connect on a single link. E.g. wired Ethernet allows for a few hundreds or a few thousands nodes in a switched network. Because of that, the difference up to 2^{64} addresses will not be used, as such they will be wasted.

5. Recommendations for implementation of variable SLAAC

This document proposes a plan to provide flexibility for implementers to now have the option to use SLAAC (Stateful Address Autoconfiguration) where previously they used DHCPv6 or static. This will also open the door to interoperability and mixed device types supporting either SLAAC, static or DHCPv6 to now be able to exist on the same subnet or VLAN without risk of interoperability issues.

It is recommended to use variable length SLAAC on network infrastructure point-to-point links as well as for host subnets where historically /64 was used that now variable length SLAAC prefix can be used up to 127 bit prefix length. It is recommended that the use of variable length prefix be based on each individual IPv6 deployment requirement. If more address space is required, necessity to break up a /64 for address space management, creating an internal hierarchical addressing plan based on prefixes delegated or allocated, then variable length prefix is now an available option in the designers toolbox that now can be utilized. Changes to DHCPv6 prefix-delegation is outside of the scope of this document.

It is recommended that ISP allocations and Customer allocations per [RFC6177] and [RFC5375] not change due to this variable SLAAC proposed standard.

6. Recommended use cases where 64 bit prefix should be utilized

Listed below are use cases where the 64 bit prefix length MUST be adhered to and in these cases variable SLAAC feature should not be utilized.

The precise 64-bit length of the interface identifier is widely mentioned in numerous RFCs describing various aspects of IPv6. It is not straightforward to distinguish cases where this has normative impact or affects interoperability. This section aims to identify specifications that contain an explicit reference to the 64-bit length. Regardless of implementation issues, the RFCs themselves would all need to be updated if the 64-bit rule was changed, even if the updates were small, which would involve considerable time and effort.

First and foremost, the RFCs describing the architectural aspects of IPv6 addressing explicitly state, refer, and repeat this apparently immutable value: Addressing Architecture [RFC4291], IPv6 Address Assignment to End Sites [RFC6177], Reserved interface identifiers [RFC5453], and ILNP Node Identifiers [RFC6741]. Customer edge routers impose /64 for their interfaces [RFC7084]. The IPv6 Subnet Model [RFC5942] points out that the assumption of a /64 prefix length is a potential implementation error.

Numerous IPv6-over-foo documents make mandatory statements with respect to the 64-bit length of the interface identifier to be used during the Stateless Autoconfiguration. These documents include [RFC2464] (Ethernet), [RFC2467] (Fiber Distributed Data Interface (FDDI)), [RFC2470] (Token Ring), [RFC2492] (ATM), [RFC2497] (ARCnet), [RFC2590] (Frame Relay), [RFC3146] (IEEE 1394), [RFC4338] (Fibre Channel), [RFC4944] (IEEE 802.15.4), [RFC5072] (PPP), [RFC5121]

[RFC5692] (IEEE 802.16), [RFC2529] (6over4), [RFC5214] (Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)), [I-D.templin-aerolink] (Asymmetric Extended Route Optimization (AERO)), [I-D.ietf-6lowpan-btle] (BLUETOOTH Low Energy), [I-D.ietf-6lo-6lobac] (IPv6 over MS/TP), and I-D.ietf-6lo-lowpanz (IPv6 packets over ITU-T G.9959).

To a lesser extent, the address configuration RFCs themselves may in some ways assume the 64-bit length of an interface identifier (e.g, [RFC4862] for the link-local addresses, DHCPv6 for the potentially assigned EUI-64-based IP addresses, and Optimistic Duplicate Address Detection [RFC4429] that computes 64-bit-based collision probabilities).

The Multicast Listener Discovery Version 1 (MLDv1) [RFC2710] and MLDv2 [RFC3810] protocols mandate that all queries be sent with a link-local source address, with the exception of MLD messages sent using the unspecified address when the link-local address is tentative [RFC3590]. At the time of publication of [RFC2710], the IPv6 addressing architecture specified link-local addresses with 64-bit interface identifiers. MLDv2 explicitly specifies the use of the fe80::/64 link-local prefix and bases the querier election algorithm on the link-local subnet prefix of length /64.

The "IPv6 Flow Label Specification" [RFC6437] gives an example of a 20-bit hash function generation, which relies on splitting an IPv6 address in two equally sized, 64-bit-length parts.

The basic transition mechanisms [RFC4213] refer to interface identifiers of length 64 for link-local addresses; other transition mechanisms such as Teredo [RFC4380] assume the use of interface identifiers of length 64. Similar assumptions are found in 6to4 [RFC3056] and 6rd [RFC5969]. Translation-based transition mechanisms such as NAT64 and NPTv6 have some dependency on prefix length, discussed below.

The proposed method [RFC7278] of extending an assigned /64 prefix from a smartphone's cellular interface to its WiFi link relies on prefix length, and implicitly on the length of the interface identifier, to be valued at 64.

The Cryptographically Generated Addresses (CGA) and Hash-Based Addresses (HBA) specifications rely on the 64-bit identifier length (see below), as do the Privacy extensions [RFC4941] and some examples in "Internet Key Exchange Version 2 (IKEv2)" [RFC7296].

464XLAT [RFC6877] explicitly mentions acquiring /64 prefixes. However, it also discusses the possibility of using the interface address on the device as the end point for the traffic, thus potentially removing this dependency.

[RFC2526] reserves a number of subnet anycast addresses by reserving some anycast interface identifiers. An anycast interface identifier so reserved cannot be less than 7 bits long. This means that a subnet prefix length longer than /121 is not possible, and a subnet of exactly /121 would be useless since all its identifiers are reserved. It also means that half of a /120 is reserved for anycast. This could of course be fixed in the way described for /127 in [RFC6164], i.e., avoiding the use of anycast within a /120 subnet. Note that support for "on-link anycast" is a standard IPv6 neighbor discovery capability [RFC4861] [RFC7094]; therefore, applications and their developers would expect it to be available.

The Mobile IP home network models [RFC4887] rely heavily on the /64 subnet length and assume a 64-bit interface identifier.

- * Multicast: [RFC3306] defines a method for generating IPv6 multicast group addresses based on unicast prefixes. This method assumes a longest prefix of 64 bits. If a longer prefix is used, there is no way to generate a specific multicast group address using this method. In such cases, the administrator would need to use an "artificial" prefix from within their allocation (a /64 or shorter) from which to generate the group address. This prefix would not correspond to a real subnet.
- * Similarly, [RFC3956], which specifies the Embedded Rendezvous Point (RP)) allowing IPv6 multicast rendezvous point addresses to be embedded in the multicast group address, would also fail, as the scheme assumes a maximum prefix length of 64 bits.
- * CGA: The Cryptographically Generated Address format [RFC3972] is heavily based on a /64 interface identifier. [RFC3972] has defined a detailed algorithm showing how to generate a 64-bit interface identifier from a public key and a 64-bit subnet prefix. Changing the /64 boundary would certainly invalidate the current CGA definition. However, the CGA might benefit in a redefined version if more bits are used for interface identifiers (which means shorter prefix length). For now, 59 bits are used for cryptographic purposes. The more bits are available, the stronger CGA could be. Conversely, longer prefixes would weaken CGA.
- * NAT64: Both stateless NAT64 [RFC6052] and stateful NAT64 [RFC6146] are flexible for the prefix length. [RFC6052] has defined multiple address formats for NAT64. In Section 2 of

"IPv4-Embedded IPv6 Address Prefix and Format" [RFC6052], the network-specific prefix could be one of /32, /40, /48, /56, /64, and /96. The remaining part of the IPv6 address is constructed by a 32-bit IPv4 address, an 8-bit u byte and a variable length suffix (there is no u byte and suffix in the case of the 96-bit Well-Known Prefix). NAT64 is therefore OK with a subnet boundary out to /96 but not longer.

- * NPTv6: IPv6-to-IPv6 Network Prefix Translation [RFC6296] is also bound to /64 boundary. NPTv6 maps a /64 prefix to another /64 prefix. When the NPTv6 Translator is configured with a /48 or shorter prefix, the 64-bit interface identifier is kept unmodified during translation. However, the /64 boundary might be changed as long as the "inside" and "outside" prefixes have the same length.
- * ILNP: Identifier-Locator Network Protocol (ILNP) [RFC6741] is designed around the /64 boundary, since it relies on locally unique 64-bit node identifiers (in the interface identifier field). While a redesign to use longer prefixes is not inconceivable, this would need major changes to the existing specification for the IPv6 version of ILNP.
- * Shim6: The Multihoming Shim Protocol for IPv6 (Shim6) [RFC5533] in its insecure form treats IPv6 addresses as opaque 128-bit objects. However, to secure the protocol against spoofing, it is essential to either use CGAs (see above) or HBAs [RFC5535]. Like CGAs, HBAs are generated using a procedure that assumes a 64-bit identifier. Therefore, in effect, secure shim6 is affected by the /64 boundary exactly like CGAs.
- * Duplicate address risk: If SLAAC was modified to work with shorter interface identifiers, the statistical risk of hosts choosing the same pseudo-random identifier [RFC7217] would increase correspondingly. The practical impact of this would range from slight to dramatic, depending on how much the interface identifier length was reduced. In particular, a /120 prefix would imply an 8-bit interface identifier and address collisions would be highly probable.
- * The link-local prefix: While [RFC4862] is careful not to define any specific length of link-local prefix within fe80::/10, the addressing architecture [RFC4291] does define the link-local interface identifier length to be 64 bits. If different hosts on a link used interface identifiers of different lengths to form a link-local address, there is potential for confusion and unpredictable results. Typically today the choice of 64 bits for the link-local interface identifier length is hard-coded per interface, in accordance with the relevant IPv6-over-foo

specification, and systems behave as if the link-local prefix was actually fe80::/64. There might be no way to change this except conceivably by manual configuration, which will be impossible if the host concerned has no local user interface.

7. Reasons for longer than 64 bit prefix length

In this section we are providing the justification for longer prefixes and shorter interface identifiers essentially variable SLAAC.

7.1. Insufficient Address Space Delegated

A site may not be delegated a sufficiently generous prefix from which to allocate a /64 prefix to all of its internal subnets. In this case, the site may either determine that it does not have enough address space to number all its network elements and thus, at the very best, be only partially operational, or it may choose to use internal prefixes longer than /64 to allow multiple subnets and the hosts within them to be configured with addresses.

In this case, the site might choose, for example, to use a /80 per subnet in combination with hosts using either manually configured addressing or DHCPv6 [RFC3315].

Scenarios that have been suggested where an insufficient prefix might be delegated include home or small office networks, vehicles, building services, and transportation services (e.g., road signs). It should be noted that the homenet architecture text [RFC7368] states that Customer Premises Equipment (CPE) should consider the lack of sufficient address space to be an error condition, rather than using prefixes longer than /64 internally.

Another scenario occasionally suggested is one where the Internet address registries actually begin to run out of IPv6 prefix space, such that operators can no longer assign reasonable prefixes to users in accordance with [RFC6177]. It is sometimes suggested that assigning a prefix such as /48 or /56 to every user site (including the smallest) as recommended by [RFC6177] is wasteful. In fact, the currently released unicast address space, 2000::/3, contains 35 trillion /48 prefixes ($(2^{45} = 35,184,372,088,832)$), of which only a small fraction have been allocated. Allowing for a conservative estimate of allocation efficiency, i.e., an HD-ratio of 0.94 [RFC4692], approximately 5 trillion /48 prefixes can be allocated. Even with a relaxed HD-ratio of 0.89, approximately one trillion /48 prefixes can be allocated. Furthermore, with only 2000::/3 currently committed for unicast addressing, we still have approximately 85% of the address space in reserve. Thus, there is no objective risk of prefix depletion by assigning /48 or /56 prefixes even to the smallest sites.

7.2. Hierarchical Addressing

Some operators have argued that more prefix bits are needed to allow an aggregated hierarchical addressing scheme within a campus or corporate network. However, if a campus or enterprise gets a /48 prefix (or shorter), then that already provides 16 bits for hierarchical allocation. In any case, flat IGP routing is widely and successfully used within rather large networks, with hundreds of routers and thousands of end systems. Therefore, there is no objective need for additional prefix bits to support hierarchy and aggregation within enterprises.

7.3. Audit Requirement

Some network operators wish to know and audit nodes that are active on a network, especially those that are allowed to communicate off-link or off-site. They may also wish to limit the total number of active addresses and sessions that can be sourced from a particular host, LAN, or site, in order to prevent potential resource-depletion attacks or other problems spreading beyond a certain scope of control. It has been argued that this type of control would be easier if only long network prefixes with relatively small numbers of possible hosts per network were used, reducing the discovery problem. However, such sites most typically operate using DHCPv6, which means that all legitimate hosts are automatically known to the DHCPv6 servers, which is sufficient for audit purposes. Such hosts could, if desired, be limited to a small range of interface identifier values without changing the /64 subnet length. Any hosts inadvertently obtaining addresses via SLAAC can be audited through Neighbor Discovery (ND) logs.

7.4. Concerns over ND Cache Exhaustion

A site may be concerned that it is open to ND cache exhaustion attacks [RFC3756], whereby an attacker sends a large number of messages in rapid succession to a series of (most likely inactive) host addresses within a specific subnet. Such an attack attempts to fill a router's ND cache with ND requests pending completion, which results in denying correct operation to active devices on the network.

One potential way to mitigate this attack would be to consider using a /120 prefix, thus limiting the number of addresses in the subnet to be similar to an IPv4 /24 prefix, which should not cause any concerns for ND cache exhaustion. Note that the prefix does need to be quite long for this scenario to be valid. The number of theoretically possible ND cache slots on the segment needs to be of the same order of magnitude as the actual number of hosts. Thus, small increases from the /64 prefix length do not have a noticeable impact; even 2^{32} potential entries, a factor of two billion decrease compared to 2^{64} , is still more than enough to exhaust the memory on current routers. Given that most link-layer mappings cause SLAAC to assume a 64-bit network boundary, in such an approach hosts would likely need to use DHCPv6 or be manually configured with addresses.

It should be noted that several other mitigations of the ND cache attack are described in [RFC6583], and that limiting the size of the cache and the number of incomplete entries allowed would also defeat the attack. For the specific case of a point-to-point link between routers, this attack is indeed mitigated by a /127 prefix [RFC6164].

7.5. Longer prefixes lengths used for embedding information

Ability to utilize the longer than 64 bit prefixes to be able to embed geographic or other information into the prefix that could be valuable to the IPv6 addressing architecture providing more flexibility to the operator.

8. Greater than 64 bit prefix usage by ISPs is strictly prohibited

ISPs SHOULD NOT send to end users prefixes in RAs that are longer than 64. This is in order to avoid a race to the bottom.

9. Comparison of Static, SLAAC, DHCPv6 and Variable SLAAC

- * Static - IPv6 address and Default Gateway:

- Pros:

- Deactivation of RA processing

- Good resistance against RA attack

Cons:

- Operational impact in configuring interface manually
- Network dynamics might require renumbering which needs work

* Static - IPv6 address and Default Route via RA

Pros:

- Does not require disabling RA processing
- Works better with FHRP router redundancy

Cons:

- RA related security issues combat with RA Guard

* DHCPv6 [RFC3315]

Pros:

- Centralized provisioning of IPv6 addressing
- IPv6, DNS, NTP can all be distributed

Cons:

- Administrative overhead of managing DHCPv6 server
- Caveats with redundancy and split scopes required for failover. Split scope and failover is resolved with DHCPv6 Failover protocol [RFC8156]
- RA related security issues combat with RA Guard

* SLAAC [RFC7217] Stable Random station-id with privacy and [RFC8064] Recommendations for Stable interface identifier

Pros:

- Automatic provisioning IPv6 address to hosts
- [RFC7217] Stable Random station-id with privacy extensions

Cons:

- RA related security issues combat with RA Guard

* Variable SLAAC with [RFC7217] Stable Random station-id with privacy and [RFC8064] Recommendations for Stable interface identifier

Pros:

- Automatic provisioning IPv6 address to hosts
- [RFC7217] Stable Random station-id with privacy extensions

Cons:

- RA related security issues combat with RA Guard
- Security is reduced with longer prefixes and shorter stable random station-id

IPv6 address deployment summary statement.

DHCPv6 [RFC3315] state machine introduces a large number of messaging packets with Normal mode, four messages called solicit, advertise, request and reply. DHCPv6 Rapid Commit mode reduces the messages from four to two messages only solicit and reply. DHCPv6 Normal mode is the Default. It is recommended to use DHCPv6 Rapid mode [RFC4039] in "high mobility" networks where clients come and go often. The overhead of four messages might not be required so two messages might enough to accommodate. However, if you have multiple DHCPv6 servers for redundancy then you need to use DHCPv6 Normal mode. If you have subnets where there are a large flat user subnets with a very large number of hosts and redundancy is required and DHCPv6 Normal mode is utilized, DHCPv6 messaging is exacerbated exponentially as the subnets flatten out further and further. As the paradigm shifts and IPv4 is eliminated as hosts subnets change to "IPv6-ONLY" subnets, the coupling of IPv4 with IPv6 Dual stack dependency is eliminated, thus removing the shackles pinning IPv6 to smaller many IPv4 subnets. This change allows IPv6 subnets to become very large and flat with the only limiting factor being the L2 switch infrastructure. In many cases Dual stacked implementations with 100's of subnets may change to a single "IPV6 ONLY" subnet. As "IPV6-ONLY" subnets will soon become the future direction of all user access infrastructure, we need a viable solution that will accommodate these very large flat subnets. The problem with DHCPv6 is that once the "M" managed bit is set to "1", all hosts on the subnet cache the Managed IP "M bit" and changes host to DHCPv6 stateful mode. Higher probability of rouge devices such as printers or other appliances misbehaving with IPv6 enabled by default, now in DHCPv6 mode, spewing of millions of DHCPv6 messages that can now impact the router control plane processing of packets. This can be alleviated with special custom Control plane policer policy, however now adds complexity and administrative overhead to DHCPv6 deployments. Enterprises and Service Providers require a viable IPv6 deployment solution that can accommodate the shortfalls of both static and DHCPv6 addressing. Static addressing due to administrative overhead of manual assignment does not provide a viable solution for even moderately sized networks. Variable SLAAC

now has the ability to fill the gaps outlined with DHCPv6 and static that can now be used as a viable ubiquitous all encompassing solution for IPv6 address deployments.

10. Variable SLAAC Use Cases

This section describes real world use cases of variable slaac that cannot be done today and with fixed 64 bit prefix lengths.

10.1. Permission-less Extension of the Network

Permission-less extensions of the network with new links (and by implication with new routers) are not supported.

The lack of possibility to realize a permission-less extension of the network is an important problem. The problem appears at the edge of the network. The permission is 'granted' for end users situated at the edge of the network. This permission is 'granted' by advertising a prefix of length 64, typically. This prefix is set in the PIO option in an RA. The end user receives this prefix, forms an address, and is able to connect to the Internet. However, the end user has no permission to further extend the network. Although s/he is able to form subsequent prefixes of a length of, say 65, and further advertise it down in the extension of the network, no other Host in that extension of the network is able to use that advertisement; a Host can not form an address with a prefix length 65 by using SLAAC. The linux error text reported in the kernel log upon reception of a plen 65 is "illegal" (or similar).

10.2. Private Networks

Private networks such as Service Provider core not accessible by customers and enterprises where all hosts are trusted are the primary use case for variable SLAAC as the shorter interface identifier does not create any security issues with not having a longer 64 bit interface identifier for privacy extensions stable interface identifier [RFC8084] due to all hosts being inherently trusted. Private internal networks such as corporate intranets traditionally have always used static IPv6 addressing for infrastructure. This manual IPv6 address assignment process for network infrastructure links can take long lead times to complete deployment. By changing the behavior of SLAAC to support variable length prefix and interface identifier allows SLAAC to be used programmatically to deploy to large scale IPv6 networks with thousands of point-to-point links. Note that network infrastructure technically does not require IPv6 addressing due to IPv6 next hop being a link local address for IGP routing protocols such as OSPF and ISIS as well as the link local address can be the peer IPv6 address for exterior gateway routing

protocols such as BGP. However for hop by hop ping and traceroute capability to have IPv6 reachability at each hop for troubleshooting jitter, latency and drops it is an IPv6 recommended best practice to configure IPv6 address on all infrastructure interfaces.

10.3. Mobile IPv6

Old MIP6 (Mobile IPv6) Working Group and old Nemo Working Group's routing solution scenarios related to Mobile IPv6 ([RFC3775]) (note: nowadays most MIP-related activity is in DMM WG) where the mobile endpoint can now obtain from the home agent variable SLAAC address and not 64 bit prefix /64 address. This maybe useful in cases where a /64 can now be managed from an addressing perspective and subdivided into blocks for manageability of MIP6 endpoints instead of allocating a single /64 per endpoint.

10.4. Home and SOHO

Home and SOHO (Small Office and Home Office) environments where internet access uses a broadband service provider single or dual homed scenario. In those such Home networking Homenet environments where HNCP (Home Network Control Protocol [RFC7788] SADR (Source Address Dependent Routing) are deployed for automatic configuration for LAN WIFI endpoint subnets can also now take advantage of variable length SLAAC in deployment scenarios. In cases where multiple routers are deployed in a home environment where routing prefix reachability needs to be advertised where Babel [RFC6126] routing protocol is utilized in those cases variable SLAAC can also be utilized to break up a /64 into multiple smaller subnets.

10.5. 3GPP V2I and V2V networking

In V2I networking (with 3GPP or with IEEE 802.11bd) the vehicle receives a /64 prefix from the cellular network (or from a Road-Side Unit). This /64 prefix can be used to form one address for the egress interface of the Mobile Router (IP On-Board Unit), but can not be used to form IP addresses for other hosts in the vehicle.

3GPP V2V networking use cases where a /56 is allocated to the 4G modem and a /64 is delegated to downstream devices within the automobile now the /64 prefixes can be sub divided into smaller prefix lengths of /65-/128. This provides additional granularity to use cases.

10.6. 6lo

6lo Working IPv6 over Network Constrained nodes working group use cases. Use cases for IoT devices where have limited network access requirements could now take advantage of variable SLAAC longer prefixes lengths /65-/128.

10.7. Large ISP's backbone POP

Large ISP backbone POPs such as IXPs where many carriers share the same backbone and ND cache exhaustion may occur due to /64 subnet size. One mitigation technique employed is the use of an ARP Sponge for IPv4 or Layer 2 multicast rate limiters for IPv6. In those particular cases a longer prefix static or variable SLAAC subnet could be utilized to reduce the maximum number of hosts on the subnet.

11. Variable SLAAC implementation

An implementation of VSLAAC (variable SLAAC) might not be particularly useful without widespread industry adoption across all major operating systems (Windows, MAC/iOS, Linux/Android, FreeBSD, and others).

There is an implementation of Variable SLAAC in the OpenBSD Operating System. In the FreeBSD OS the stack does not accept a plen different than 64; an issue was reported to the OS maintainers.

There is an implementation for Variable SLAAC for the linux Operating System. This implementation is available freely at <https://github.com/dmytroshyti/variable-slaac>. It was submitted for consideration to the linux OS stack maintainers. A few emails of feedback were received.

The linux implementation for Variable SLAAC contains a parameter that can be controlled in the command line (a sysctl). This parameter has two potential values: 0 and 1; by default it is set to 0. The value of 0 means that the stack acts as previously: it does not accept a prefix of a length other than 64 for the SLAAC process. The value of 1 makes that prefixes of lengths other than 64 are accepted for the SLAAC mechanism of forming addresses.

In case the sysctl is set to 1 (by default it is 0), this draft precludes the use of EUI64 based, 64 bit fixed length interface identifier generation algorithms, and allows the use of any standard variable interface identifier generation algorithm for the auto generating variable length interface identifier less than 64 bits, for example [RFC4941] Privacy Extensions for Stateless Address

Autoconfiguration in IPv6 or [RFC7217] Semantically opaque interface identifier with SLAAC privacy extension algorithm for stable variable length interface identifier per [RFC8064]. In this particular case the prefix will be greater than 64 bits and the stable interface identifier will be less than 64 bits in length.

In case the `sysctl` is set to 1 (by default it is 0) this draft precludes the use of EUI64 based, 64 bit fixed length interface identifier generation algorithms, and allows the use of any standard variable interface identifier generation algorithm for the auto generating variable length interface identifier greater than 64 bits for example [RFC4941] Privacy Extensions for Stateless Address Autoconfiguration in IPv6 or [RFC7217] Semantically opaque interface identifier with SLAAC privacy extension algorithm for stable variable length interface identifier per [RFC8064]. In this particular case the prefix will be less than 64 bits and the stable interface identifier will be less than 64 bits.

Draft `rfc4941bis` Privacy Extension for SLAAC using [RFC4086] Pseudo-Random Number Generator (PRNG) can also be used as a possible method of generating greater than 64 bit or less than 64 bit interface identifier automatically since stated in the draft that the interface identifier can be generated for any arbitrary length.
<https://datatracker.ietf.org/doc/draft-ietf-6man-rfc4941bis/>

12. Applicability Statements

The `sysctl` parameter for variable length interface identifier is designed to allow administrators to send variable length prefixes in the PIO list advertisement to the host and hosts supporting this variable interface identifier option would be able to process and use the prefix with variable interface identifier in the PIO list.

13. Router and Operational Considerations

Default IPv6 routers that might need to send prefixes in RAs of length different than 64 MAY do so.

14. Host Behavior Considerations

Host operating system support will be backwards compatible. The hosts that do not support Variable SLAAC will not accept RAs with plens different than 64 and will not accept the manual configuration of addresses with prefix lengths different than 64. Hosts that support Variable SLAAC MUST set the respective `sysctl` parameter to 1. This parameter is reset (set to 0) by default.

15. Security Considerations

The administrator should be aware to maintain 64 bit interface identifier for privacy when connected directly to the internet so that entropy for optimal heuristics are maintained for security.

Variable length interface identifier shorter than 64 bits should be used within networks where there are out-of-band guarantees that the hosts are trusted (e.g. corporate intranets and private networks).

16. IANA Considerations

No request to IANA.

17. Contributors

Contributors.

18. Acknowledgements

Maciej Zenczykowski (Żenczykowski), Kernel Networking Developer at Google provided feedback about VSLAAC in linux, and expressed doubts.

19. References

19.1. Normative References

[I-D.bourbaki-6man-classless-ipv6]

Bourbaki, N., "IPv6 is Classless", Work in Progress, Internet-Draft, draft-bourbaki-6man-classless-ipv6-06, 18 April 2022, <<https://www.ietf.org/internet-drafts/draft-bourbaki-6man-classless-ipv6-06.txt>>.

[I-D.ietf-6lo-6lobac]

Lynn, K., Martocci, J., Neilson, C., and S. Donaldson, "Transmission of IPv6 over Master-Slave/Token-Passing (MS/TP) Networks", Work in Progress, Internet-Draft, draft-ietf-6lo-6lobac-08, 13 March 2017, <<https://www.ietf.org/archive/id/draft-ietf-6lo-6lobac-08.txt>>.

[I-D.ietf-6lowpan-btle]

Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "Transmission of IPv6 Packets over BLUETOOTH Low Energy", Work in Progress, Internet-Draft, draft-ietf-6lowpan-btle-12, 12 February 2013, <<https://www.ietf.org/archive/id/draft-ietf-6lowpan-btle-12.txt>>.

[I-D.mishra-v6ops-variable-slaac-problem-stmt]

Mishra, G., Petrescu, A., Kottapalli, N., Mudric, D., and D. Shyti, "SLAAC with prefixes of arbitrary length in PIO (Variable SLAAC) - A Problem Statement", Work in Progress, Internet-Draft, draft-mishra-v6ops-variable-slaac-problem-stmt-03, 20 January 2022, <<https://www.ietf.org/archive/id/draft-mishra-v6ops-variable-slaac-problem-stmt-03.txt>>.

[I-D.templin-aerolink]

Templin, F. L., "Asymmetric Extended Route Optimization (AERO)", Work in Progress, Internet-Draft, draft-templin-aerolink-82, 10 May 2018, <<https://www.ietf.org/archive/id/draft-templin-aerolink-82.txt>>.

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

[RFC2450] Hinden, R., "Proposed TLA and NLA Assignment Rule", RFC 2450, DOI 10.17487/RFC2450, December 1998, <<https://www.rfc-editor.org/info/rfc2450>>.

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

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

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

- [RFC2492] Armitage, G., Schuster, P., and M. Jork, "IPv6 over ATM Networks", RFC 2492, DOI 10.17487/RFC2492, January 1999, <<https://www.rfc-editor.org/info/rfc2492>>.
- [RFC2497] Souvatzis, I., "Transmission of IPv6 Packets over ARCnet Networks", RFC 2497, DOI 10.17487/RFC2497, January 1999, <<https://www.rfc-editor.org/info/rfc2497>>.
- [RFC2526] Johnson, D. and S. Deering, "Reserved IPv6 Subnet Anycast Addresses", RFC 2526, DOI 10.17487/RFC2526, March 1999, <<https://www.rfc-editor.org/info/rfc2526>>.
- [RFC2529] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529, DOI 10.17487/RFC2529, March 1999, <<https://www.rfc-editor.org/info/rfc2529>>.
- [RFC2590] Conta, A., Malis, A., and M. Mueller, "Transmission of IPv6 Packets over Frame Relay Networks Specification", RFC 2590, DOI 10.17487/RFC2590, May 1999, <<https://www.rfc-editor.org/info/rfc2590>>.
- [RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, DOI 10.17487/RFC2710, October 1999, <<https://www.rfc-editor.org/info/rfc2710>>.
- [RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, DOI 10.17487/RFC3056, February 2001, <<https://www.rfc-editor.org/info/rfc3056>>.
- [RFC3146] Fujisawa, K. and A. Onoe, "Transmission of IPv6 Packets over IEEE 1394 Networks", RFC 3146, DOI 10.17487/RFC3146, October 2001, <<https://www.rfc-editor.org/info/rfc3146>>.
- [RFC3177] IAB and IESG, "IAB/IESG Recommendations on IPv6 Address Allocations to Sites", RFC 3177, DOI 10.17487/RFC3177, September 2001, <<https://www.rfc-editor.org/info/rfc3177>>.
- [RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, DOI 10.17487/RFC3306, August 2002, <<https://www.rfc-editor.org/info/rfc3306>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<https://www.rfc-editor.org/info/rfc3315>>.

- [RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, DOI 10.17487/RFC3513, April 2003, <<https://www.rfc-editor.org/info/rfc3513>>.
- [RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, DOI 10.17487/RFC3587, August 2003, <<https://www.rfc-editor.org/info/rfc3587>>.
- [RFC3590] Haberman, B., "Source Address Selection for the Multicast Listener Discovery (MLD) Protocol", RFC 3590, DOI 10.17487/RFC3590, September 2003, <<https://www.rfc-editor.org/info/rfc3590>>.
- [RFC3627] Savola, P., "Use of /127 Prefix Length Between Routers Considered Harmful", RFC 3627, DOI 10.17487/RFC3627, September 2003, <<https://www.rfc-editor.org/info/rfc3627>>.
- [RFC3756] Nikander, P., Ed., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, DOI 10.17487/RFC3756, May 2004, <<https://www.rfc-editor.org/info/rfc3756>>.
- [RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, DOI 10.17487/RFC3775, June 2004, <<https://www.rfc-editor.org/info/rfc3775>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", RFC 3956, DOI 10.17487/RFC3956, November 2004, <<https://www.rfc-editor.org/info/rfc3956>>.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <<https://www.rfc-editor.org/info/rfc3972>>.
- [RFC4039] Park, S., Kim, P., and B. Volz, "Rapid Commit Option for the Dynamic Host Configuration Protocol version 4 (DHCPv4)", RFC 4039, DOI 10.17487/RFC4039, March 2005, <<https://www.rfc-editor.org/info/rfc4039>>.

- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, DOI 10.17487/RFC4213, October 2005, <<https://www.rfc-editor.org/info/rfc4213>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4338] DeSanti, C., Carlson, C., and R. Nixon, "Transmission of IPv6, IPv4, and Address Resolution Protocol (ARP) Packets over Fibre Channel", RFC 4338, DOI 10.17487/RFC4338, January 2006, <<https://www.rfc-editor.org/info/rfc4338>>.
- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <<https://www.rfc-editor.org/info/rfc4380>>.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, DOI 10.17487/RFC4429, April 2006, <<https://www.rfc-editor.org/info/rfc4429>>.
- [RFC4548] Gray, E., Rutenmiller, J., and G. Swallow, "Internet Code Point (ICP) Assignments for NSAP Addresses", RFC 4548, DOI 10.17487/RFC4548, May 2006, <<https://www.rfc-editor.org/info/rfc4548>>.
- [RFC4692] Huston, G., "Considerations on the IPv6 Host Density Metric", RFC 4692, DOI 10.17487/RFC4692, October 2006, <<https://www.rfc-editor.org/info/rfc4692>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.

- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC4887] Thubert, P., Wakikawa, R., and V. Devarapalli, "Network Mobility Home Network Models", RFC 4887, DOI 10.17487/RFC4887, July 2007, <<https://www.rfc-editor.org/info/rfc4887>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<https://www.rfc-editor.org/info/rfc4944>>.
- [RFC5072] Varada, S., Ed., Haskins, D., and E. Allen, "IP Version 6 over PPP", RFC 5072, DOI 10.17487/RFC5072, September 2007, <<https://www.rfc-editor.org/info/rfc5072>>.
- [RFC5121] Patil, B., Xia, F., Sarikaya, B., Choi, JH., and S. Madanapalli, "Transmission of IPv6 via the IPv6 Convergence Sublayer over IEEE 802.16 Networks", RFC 5121, DOI 10.17487/RFC5121, February 2008, <<https://www.rfc-editor.org/info/rfc5121>>.
- [RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, DOI 10.17487/RFC5214, March 2008, <<https://www.rfc-editor.org/info/rfc5214>>.
- [RFC5375] Van de Velde, G., Popoviciu, C., Chown, T., Bonness, O., and C. Hahn, "IPv6 Unicast Address Assignment Considerations", RFC 5375, DOI 10.17487/RFC5375, December 2008, <<https://www.rfc-editor.org/info/rfc5375>>.
- [RFC5453] Krishnan, S., "Reserved IPv6 Interface Identifiers", RFC 5453, DOI 10.17487/RFC5453, February 2009, <<https://www.rfc-editor.org/info/rfc5453>>.
- [RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, DOI 10.17487/RFC5533, June 2009, <<https://www.rfc-editor.org/info/rfc5533>>.

- [RFC5535] Bagnulo, M., "Hash-Based Addresses (HBA)", RFC 5535, DOI 10.17487/RFC5535, June 2009, <<https://www.rfc-editor.org/info/rfc5535>>.
- [RFC5692] Jeon, H., Jeong, S., and M. Riegel, "Transmission of IP over Ethernet over IEEE 802.16 Networks", RFC 5692, DOI 10.17487/RFC5692, October 2009, <<https://www.rfc-editor.org/info/rfc5692>>.
- [RFC5942] Singh, H., Beebee, W., and E. Nordmark, "IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes", RFC 5942, DOI 10.17487/RFC5942, July 2010, <<https://www.rfc-editor.org/info/rfc5942>>.
- [RFC5969] Townsley, W. and O. Troan, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) -- Protocol Specification", RFC 5969, DOI 10.17487/RFC5969, August 2010, <<https://www.rfc-editor.org/info/rfc5969>>.
- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <<https://www.rfc-editor.org/info/rfc6052>>.
- [RFC6126] Chroboczek, J., "The Babel Routing Protocol", RFC 6126, DOI 10.17487/RFC6126, April 2011, <<https://www.rfc-editor.org/info/rfc6126>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC6164] Kohno, M., Nitzan, B., Bush, R., Matsuzaki, Y., Colitti, L., and T. Narten, "Using 127-Bit IPv6 Prefixes on Inter-Router Links", RFC 6164, DOI 10.17487/RFC6164, April 2011, <<https://www.rfc-editor.org/info/rfc6164>>.
- [RFC6177] Narten, T., Huston, G., and L. Roberts, "IPv6 Address Assignment to End Sites", BCP 157, RFC 6177, DOI 10.17487/RFC6177, March 2011, <<https://www.rfc-editor.org/info/rfc6177>>.
- [RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <<https://www.rfc-editor.org/info/rfc6296>>.

- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<https://www.rfc-editor.org/info/rfc6437>>.
- [RFC6583] Gashinsky, I., Jaeggli, J., and W. Kumari, "Operational Neighbor Discovery Problems", RFC 6583, DOI 10.17487/RFC6583, March 2012, <<https://www.rfc-editor.org/info/rfc6583>>.
- [RFC6741] Atkinson, RJ. and SN. Bhatti, "Identifier-Locator Network Protocol (ILNP) Engineering Considerations", RFC 6741, DOI 10.17487/RFC6741, November 2012, <<https://www.rfc-editor.org/info/rfc6741>>.
- [RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", RFC 6877, DOI 10.17487/RFC6877, April 2013, <<https://www.rfc-editor.org/info/rfc6877>>.
- [RFC7084] Singh, H., Beebee, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, DOI 10.17487/RFC7084, November 2013, <<https://www.rfc-editor.org/info/rfc7084>>.
- [RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil, "Architectural Considerations of IP Anycast", RFC 7094, DOI 10.17487/RFC7094, January 2014, <<https://www.rfc-editor.org/info/rfc7094>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<https://www.rfc-editor.org/info/rfc7217>>.
- [RFC7278] Byrne, C., Drown, D., and A. Vizdal, "Extending an IPv6 /64 Prefix from a Third Generation Partnership Project (3GPP) Mobile Interface to a LAN Link", RFC 7278, DOI 10.17487/RFC7278, June 2014, <<https://www.rfc-editor.org/info/rfc7278>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.

- [RFC7368] Chown, T., Ed., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", RFC 7368, DOI 10.17487/RFC7368, October 2014, <<https://www.rfc-editor.org/info/rfc7368>>.
- [RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <<https://www.rfc-editor.org/info/rfc7421>>.
- [RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April 2016, <<https://www.rfc-editor.org/info/rfc7788>>.
- [RFC8064] Gont, F., Cooper, A., Thaler, D., and W. Liu, "Recommendation on Stable IPv6 Interface Identifiers", RFC 8064, DOI 10.17487/RFC8064, February 2017, <<https://www.rfc-editor.org/info/rfc8064>>.
- [RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <<https://www.rfc-editor.org/info/rfc8084>>.
- [RFC8156] Mrugalski, T. and K. Kinnear, "DHCPv6 Failover Protocol", RFC 8156, DOI 10.17487/RFC8156, June 2017, <<https://www.rfc-editor.org/info/rfc8156>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

19.2. Informative References

- [RFC8273] Brzozowski, J. and G. Van de Velde, "Unique IPv6 Prefix per Host", RFC 8273, DOI 10.17487/RFC8273, December 2017, <<https://www.rfc-editor.org/info/rfc8273>>.

Appendix A. ChangeLog

The changes are listed in reverse chronological order, most recent changes appearing at the top of the list.

-02: removed the flag from the RA, and added a parameter (sysctl) in the linux implementation.

-01:

-00: initial version.

Authors' Addresses

Gyan Mishra
Verizon Inc.
Email: gyan.s.mishra@verizon.com

Alexandre Petrescu
CEA, LIST
CEA Saclay
91190 Gif-sur-Yvette
France
Phone: +33169089223
Email: Alexandre.Petrescu@cea.fr

Naveen Kottapalli
Benu Networks
300 Concord Road
Billerica, MA 01821
United States of America
Phone: +1 978 223 4700
Email: nkottapalli@benu.net

Dusan Mudric
Ciena
Canada
Phone: +1-613-670-2425
Email: dmudric@ciena.com

Dmytro Shyti
SFR
Paris
France
Email: dmytro.shyti@sfr.com

SPRING
Internet-Draft
Intended status: Informational
Expires: January 10, 2022

W. Cheng
China Mobile
C. Xie
China Telecom
R. Bonica
Juniper
D. Dukes
Cisco Systems
C. Li
Huawei
P. Shaofu
ZTE
W. Henderickx
Nokia
July 09, 2021

Compressed SRv6 SID List Requirements
draft-srcompdt-spring-compression-requirement-07

Abstract

This document specifies requirements for solutions to compress SRv6 SID lists.

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 10, 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 publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions used in this document	4
2.1. Requirements Language	4
2.2. Terminology	4
3. SRv6 SID List Compression Requirements	4
3.1. Dataplane Efficiency and Performance Requirements	4
3.1.1. Encapsulation Header Size	5
3.1.2. Forwarding Efficiency	5
3.1.3. State Efficiency	6
4. SRv6 Specific Requirements	6
4.1. SRv6 Based	6
4.2. Functional Requirements	7
4.2.1. SRv6 Functionality	7
4.2.2. Heterogeneous SID lists	8
4.2.3. SID list length	8
4.2.4. SID summarization	8
4.3. Operational Requirements	9
4.3.1. Lossless Compression	9
4.3.2. Preservation of non-routing information	9
4.3.3. Address Planning	10
4.4. Scalability Requirements	10
4.4.1. Adjacency segment scale	10
4.4.2. Prefix segment scale	11
4.4.3. Service Scale	11
4.4.4. Compression Levels	11
5. Protocol Design Requirements	11
5.1. SRv6 Base Coexistence	11
6. Security Requirements	12
6.1. Security Mechanisms	12
6.2. SR Domain Protection	12
7. IANA Considerations	12
8. Security Considerations	13
9. Contributors	13
10. Normative References	13
Appendix A. Proposed Requirements	14
A.1. IPv6 Based	14

A.2. Point to Multipoint	15
A.3. Parsability	15
Authors' Addresses	15

1. Introduction

The SPRING working group defined SRv6, with [RFC8402] describing how the Segment Routing (SR) architecture is instantiated on two data-planes: SR over MPLS (SR-MPLS) and SR over IPv6 (SRv6). SRv6 uses a routing header called the SR Header (SRH) [RFC8754] and defines SRv6 SID behaviors and a registry for identifying them in [RFC8986]. SRv6 is a proposed standard and is deployed today.

The SPRING working group has observed that some use cases, such as strict path TE, may require long SRv6 SID lists. There are several proposed methods to reduce the resulting SRv6 encapsulation size by compressing the SID list.

The SPRING working group formed a design team to define requirements for, and analyze proposals to, compress SRv6 SID lists.

It is a goal of the design team to identify solutions to SRv6 SID list compression that are based on the SRv6 standards. As such, this document provides requirements for SRv6 SID list compression solutions that utilize the existing SRv6 data plane and control plane.

It is also a goal of the design team to consider proposals that are not based on the SRv6 data plane and control plane. As such, this document includes requirements to evaluate whether a compression proposal provides all the functionality of SRv6 (section "SRv6 Functionality") in addition to satisfying compression specific requirements.

For each requirement, a description, rationale and metrics are described.

The design team will produce a separate document to analyze the proposals.

This document is a draft; additional requirements are under review, additional requirements will be added, and current requirements may change. Appendix A contains a subset of requirements without unanimous consensus. Additional requirements without unanimous consensus are not in the appendix.

2. Conventions used in this document

2.1. Requirements Language

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

2.2. Terminology

SR: Segment Routing

SRH: Segment Routing Header

MPLS: Multiprotocol Label Switching

SR-MPLS: Segment Routing over MPLS data plane

SID: Segment Identifier

SRv6: Segment Routing over IPv6

SRv6 SID List: A list of SRv6 SIDs

Compression proposal: A proposal to compress SRv6 SID lists

SRv6 base: SRv6 as defined in [RFC8402], [RFC8754], [RFC8986]

SID numbering space: may be implemented as

- o a single IGP instance
- o a single IGP level or area
- o two or more autonomous systems that coordinate SID numbering space
- o two or more IGP instances that coordinate SID numbering space

SRv6 Encapsulation Header: The IPv6 header, and any extension headers preceding a payload, used to implement a SRv6 base or compression proposal.

3. SRv6 SID List Compression Requirements

3.1. Dataplane Efficiency and Performance Requirements

3.1.1. Encapsulation Header Size

Description: The compression proposal MUST reduce the size of the SRv6 encapsulation header.

Rationale: A smaller SRv6 encapsulation results in better MTU efficiency.

Metric: Compression is the ratio of the IPv6 encapsulation size of SRv6 as defined in [RFC8402], [RFC8754], [RFC8986] vs the IPv6 encapsulation size of a given proposal. The encapsulation savings of a compression proposal vs the SRv6 base is a useful measurement to compare proposals.

The encapsulation metric (E) records the number of bytes required for a proposal to encapsulate a packet given a specific segment list.

o $E(\text{proposal}, \text{segment list})$.

The encapsulation savings (ES) records the encapsulation savings for a proposal to encapsulate a packet given a specific segment list.

o $ES(\text{proposal}, \text{segment list}) = 1 - E(\text{proposal}, \text{segment list})/E(\text{SRv6 base}, \text{segment list})$.

3.1.2. Forwarding Efficiency

Description: The compression proposal SHOULD minimize the number of required hardware resources accessed to process a segment.

Rationale: Efficiency in bits on the wire and processing efficiency are both important. Optimizing one at the expense of the other may lead to significant performance impact.

Metric: The data plane efficiency metric (D) records the data plane forwarding efficiency of the proposed solution. Two metrics are used and recorded at each segment endpoint:

- o $D.PRS(\text{segment list})$: number of headers parsed during processing of the segment list, starting from and including the IPv6 header.
- o $D.LKU(\text{segment list})$: number of FIB lookups during processing of the segment list. The type of lookup is also recorded as longest prefix match (LPM) or exact match (EM)

3.1.3. State Efficiency

Description: The compression proposal SHOULD minimize the amount of additional forwarding state stored at a node.

Rationale: Additional state increases the complexity of the control plane and data plane. It can also result in an increase in memory usage.

Metric: The state efficiency metric (S) records the amount of additional forwarding state required by the proposed solution.

- o S(node parameters): the number of additional forwarding states that need to be stored at a node, given a set of node parameters consisting of the number of nodes in the network, number of local interfaces, number of adjacencies. The forwarding state is counted as entries required in a Forwarding Information Base (FIB) at a node.

4. SRv6 Specific Requirements

4.1. SRv6 Based

Description: A solution to compress SRv6 SID Lists SHOULD be based on the SRv6 architecture, control plane and data plane. The compression solution MAY be based on a different data plane and control plane, provided that it derives sufficient benefit.

Rationale: A compression proposal built on existing IETF standards is preferable to creating new standards with equivalent functionality and performance.

Metric: The utilization metric (U) records whether a proposal utilizes the SRv6 specifications.

Utilization is recorded in a table, with a column per proposal and rows consisting of the following metrics:

- o U.RFC8402: utilizes [RFC8402].
- o U.RFC8754: utilizes [RFC8754].
- o U.PGM: utilizes [RFC8986].
- o U.IGP: utilizes [I-D.ietf-lsr-isis-srv6-extensions].
- o U.BGP: utilizes [I-D.ietf-bess-srv6-services].
- o U.POL: utilizes [I-D.ietf-spring-segment-routing-policy].
- o U.BLS: utilizes [I-D.ietf-idr-bgppls-srv6-ext].
- o U.SVC: utilizes [I-D.ietf-spring-sr-service-programming].
- o U.OAM: utilizes [I-D.ietf-6man-spring-srv6-oam].
- o U.ALG: utilizes [I-D.ietf-lsr-flex-algo].

Each cell contains "yes" for utilizes, or "no" for does not utilize.

4.2. Functional Requirements

4.2.1. SRv6 Functionality

Description: A solution to compress an SRv6 SID list MUST support the functionality of SRv6. This requirement ensures no SRv6 functionality is lost. It is particularly important to understand how a proposal, as evaluated in section "SRv6 Based", provides this functionality.

Rationale: Operators require SRv6 functionality. Evaluating the extent to which a proposal supports SRv6 functionality is important for operators and implementors to understand the impact on network operations.

Metric: The Functionality metric (F) records whether a proposal supports SRv6 functionality and how the functionality is provided.

Functionality is recorded in a table with columns for each proposal and rows consisting of the following metrics:

- o F.SID: Supports SRv6 SID functionality as described in [RFC8402]
- o F.SCOPE: Supports globally and locally scoped SID functionality as described in [RFC8402]
- o F.PFX: Supports prefix SID functionality as described in [RFC8402] and [RFC8986]
- o F.ADJ: Supports adjacency SID functionality as described in [RFC8402] and [RFC8986]
- o F.BIND: Supports binding SID functionality as described in [RFC8402] and [RFC8986]
- o F.PEER: Supports BGP peering SID functionality as described in [RFC8402] and [RFC8986]
- o F.SVC: Supports L3 and L2 VPN service SID functionality as described in [RFC8986]
- o F.ALG: Supports flexible algorithms functionality as described in [I-D.ietf-lsr-flex-algo]
- o F.TILFA: Supports TI-LFA functionality as described in [I-D.ietf-rtgwg-segment-routing-ti-lfa]
- o F.SEC: Supports securing an SR domain with ingress filtering as functionally defined in [RFC8754]
- o F.IGP: Supports distributing topological SIDs and behaviors via ISIS as functionally described in [I-D.ietf-lsr-isis-srv6-extensions]
- o F.BGP: Supports BGP VPNs as functionally described in [I-D.ietf-bess-srv6-services]

- o F.POL: Supports SR policies and steering traffic over those policies as functionally described in [I-D.ietf-spring-segment-routing-policy]
- o F.BLS: Supports Link State distribution via BGP as functionally described in [I-D.ietf-idr-bgpls-srv6-ext]
- o F.SFC: Supports stateless service programming as functionally described in [I-D.ietf-spring-sr-service-programming]
- o F.PING: Supports pinging a SID to verify the SID is implemented as functionally described in [I-D.ietf-6man-spring-srv6-oam]

Each cell contains the specification name documenting the functionality.

4.2.2. Heterogeneous SID lists

Description: The compression proposal SHOULD support a combination of compressed and non-compressed segments in a single path. As an example, a solution may satisfy this requirement without being SRv6 based by using a binding SID to impose an additional SRv6 header (IPv6 header plus optional SRH) with non-compressed SID.

Rationale: Support of SID lists with compressed and non-compressed SIDs reduces encapsulation size when not all SRv6 nodes deploy the compression proposal or 128-bit SIDs are required.

Metric: A compliant compression proposal supports both:

- o classic 128-bit SRv6 SIDs in the IPv6 Destination Address field
- o segment lists (i.e., paths) with both compressed and 128-bit SRv6 SIDs.

4.2.3. SID list length

Description: The compression proposal MUST be able to represent SR paths that contain up to 16 segments.

Rationale: Strict TE paths require SID list lengths proportional to the diameter of the SR domain.

Metric: The compression proposal must be able to steer a packet through an SR path that contains up to sixteen segments.

4.2.4. SID summarization

Description: The solution MUST be compatible with segment summarization.

Rationale: Summarization of segments is a key benefit of SRv6 vs SR MPLS. In interdomain deployments, any node can reach any other node via a single prefix segment. Without summarization, border router SIDs must be leaked, and an additional global prefix segment is required for each domain border to be traversed.

Metric: A solution supports summarization when segments can be summarized for advertisement into other IGP domains or levels.

4.3. Operational Requirements

4.3.1. Lossless Compression

Description: A path traversed using a compressed SID list MUST always be the same as the path traversed using the uncompressed SID list if no compression was applied.

Rationale: In SRv6, we can represent a path to meet certain objectives. A compression proposal needs to support the objectives with the same path.

Metric: Information present in the pre-compression segment list MUST also be present in the post-compression SID list.

4.3.2. Preservation of non-routing information

Description: The compression mechanism MUST NOT cause the loss of non-routing information when delivering a packet from the SR ingress node to the egress/penultimate SR node

Rationale: SRv6 ingress nodes encode non-routing information in the IPv6 header chain. This information can be encoded in the following fields:

- o Hop Count
- o DSCP bits
- o ECN bits
- o Flow label
- o HBH Options Extension header
- o Fragment Extension header
- o Authentication Extension header
- o Encrypted Security Payload Extension header
- o Destination Options Extension header

Some of these fields are mutable (e.g., Hop Count) while others are immutable (e.g., Fragment Extension Header).

Some of these fields contain information that is required by every node along a packet's delivery path (e.g., Hop Count). Others contain information that is required only by the packet's ultimate destination (e.g., Fragment Extension Header).

Therefore, the compression mechanism MUST NOT prevent this information from being delivered, in an IPv6 header chain, to any node that needs it.

Metric: The SR source node encapsulates its payload (e.g., Ethernet, IP, TCP) in an IPv6 header. The SRv6 header contains both routing and non-routing information. The compression mechanism MUST NOT cause the loss of non-routing information when delivering a packet from the SR ingress node to the egress/penultimate SR node.

4.3.3. Address Planning

Description: Network operators require addressing plan flexibility, The compression mechanism MUST support flexible IPv6 address planning, it MUST support deployment by using GUA from different address blocks.

Rationale: The address planning of the network may vary based on the addressing scheme of the operator, so the solution MUST support a flexible addressing scheme. Operators need to deploy the solution based on their own address planning.

Metric: The compression proposal supports locators drawn from different prefixes with the solutions analysis indicating efficiency.

4.4. Scalability Requirements

4.4.1. Adjacency segment scale

Description: The compression proposal MUST be capable of representing 65000 adjacency segments per node

Rationale: Typically, network operators deploy networks with tens or hundreds of adjacency segments per node, but some network operators may deploy networks that use more adjacency segments per node.

Metric: A proposal that allows 65000 adjacency segments per node satisfies this requirement.

4.4.2. Prefix segment scale

Description: The compression proposal MUST be capable of representing 1 million prefix segments per SID numbering space.

Rationale: Typically, network operators deploy networks with thousands of prefix segments per SID numbering space, but some network operators may deploy networks that use more prefix segments per SID numbering space.

Metric: A proposal that allows 1 million prefix segments per SID numbering space satisfies this requirement.

4.4.3. Service Scale

Description: The compression proposal MUST be capable of representing 1 million services per node.

Rationale: Typically, network operators deploy networks with tens to hundreds of thousands of services per node, but some network operators may deploy networks that use more services per node.

Metric: A proposal that allows 1 million services per node satisfies this requirement.

4.4.4. Compression Levels

Description: The compression proposal SHOULD be able to support multiple levels of compression.

Rationale: The compression proposal will be deployed in networks of varying size with SID numbering spaces of varying size. Network and service scale can directly impact SID length and the ability of a proposal to compress the SID list.

Metric: A compression proposal that supports relatively better compression for smaller SID numbering spaces and service scale satisfies this requirement.

5. Protocol Design Requirements

5.1. SRv6 Base Coexistence

Description: The compression proposal MUST support simultaneous deployment with SRv6 networks.

Rationale: SRv6 is deployed today. A compression proposal that interoperates well with SRv6, as deployed, will reduce the overhead

and simplify operations. For Network operators who would migrate to compressed SRv6 SID lists, the migration is expected to gradually occur over a period of time as they upgrade networks, domains, device families and software instances.

Metric: A compliant compression proposal provides the following

- o Supports simultaneous deployment at a node with current SRv6 SIDs.
- o Supports simultaneous deployment at a node with current SRv6 control plane.
- o Supports simultaneous operation of current SRv6 paths with compressed paths.
- o Supports the behaviors in [RFC8986].
- o Does not require removal of existing IPv6 address planning.

6. Security Requirements

6.1. Security Mechanisms

Description: The compression solution SHOULD be able to address security issues that it introduces, using existing security mechanisms.

Rationale: It is important to identify security issues and how to address them in any specification.

Metric: A compression solution that does not introduce unresolved security issues meets this requirement.

6.2. SR Domain Protection

Description: A compression solution must not require nodes outside the SR domain to know SID values within the SR domain, and it must provide the ability to block nodes outside an SR domain from accessing SIDs.

Rationale: The unauthorized use of SIDs within the SR domain by nodes outside the domain can disrupt an operators' network.

Metric: A compliant solution describes how access to SIDs within the SR domain is denied to nodes outside the SR domain.

7. IANA Considerations

This document has no requests to IANA.

8. Security Considerations

TBD

9. Contributors

The following individuals contributed to this draft

Sanders Steffann, SJM Steffann Consultancy, sander@steffann.nl

10. Normative References

[I-D.ietf-6man-spring-srv6-oam]

Ali, Z., Filsfils, C., Matsushima, S., Voyer, D., and M. Chen, "Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6)", draft-ietf-6man-spring-srv6-oam-10 (work in progress), April 2021.

[I-D.ietf-bess-srv6-services]

Dawra, G., Filsfils, C., Talaulikar, K., Raszuk, R., Decraene, B., Zhuang, S., and J. Rabadan, "SRv6 BGP based Overlay Services", draft-ietf-bess-srv6-services-07 (work in progress), April 2021.

[I-D.ietf-idr-bgppls-srv6-ext]

Dawra, G., Filsfils, C., Talaulikar, K., Chen, M., Bernier, D., and B. Decraene, "BGP Link State Extensions for SRv6", draft-ietf-idr-bgppls-srv6-ext-07 (work in progress), March 2021.

[I-D.ietf-lsr-flex-algo]

Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", draft-ietf-lsr-flex-algo-15 (work in progress), April 2021.

[I-D.ietf-lsr-isis-srv6-extensions]

Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extension to Support Segment Routing over IPv6 Dataplane", draft-ietf-lsr-isis-srv6-extensions-14 (work in progress), April 2021.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]

Litkowski, S., Bashandy, A., Filsfils, C., Francois, P., Decraene, B., and D. Voyer, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-06 (work in progress), February 2021.

- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-11 (work in progress), April 2021.
- [I-D.ietf-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., Bernier, D., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Service Programming with Segment Routing", draft-ietf-spring-sr-service-programming-04 (work in progress), March 2021.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.
- [RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <<https://www.rfc-editor.org/info/rfc8986>>.

Appendix A. Proposed Requirements

This appendix contains requirements that the design team discussed but could not be agreed upon.

A.1. IPv6 Based

Description: The compression mechanism requires every node along the packet's delivery path to be IPv6-capable. It MUST not require any

node along the packet's forwarding path to support any other forwarding plane (e.g., IPv4, MPLS)

Rational: According to RFC 8402, SRv6 is an instantiation of the SR Architecture over the IPv6 data plane.

Metric: A compliant solution requires every node along the packet's delivery path to be IPv6-capable. It does not require any node along the packet's forwarding path to support any other forwarding plane (e.g., IPv4, MPLS)

A.2. Point to Multipoint

Description: The compression mechanism SHOULD support point-to-multipoint SR paths.

Rationale: Many VPN services require point-to-multipoint SR paths.

Metric: A compliant proposal can encode a multicast address in the ultimate segment of the segment list.

A.3. Parsability

Description: The compression mechanism MUST be parsable. That is, the node that consumes the compressed SID list must be able to decode the active and next segment. Parsing information MAY be conveyed in either the forwarding or control plane.

Rationale: Failure to parse the compressed SID list leads to undesired behaviors.

Metric: In the nominal case the producer and consumer of the SID list agree on the active segment and next segment. In foreseeable failure modes it is possible to determine why the producer and consumer don't agree.

Authors' Addresses

Weiqiang Cheng
China Mobile

Email: chengweiqiang@chinamobile.com

Chongfeng Xie
China Telecom

Email: xiechf@chinatelecom.cn

Ron Bonica
Juniper

Email: rbonica@juniper.net

Darren Dukes
Cisco Systems

Email: ddukes@cisco.com

Cheng Li
Huawei

Email: c.l@huawei.com

Peng Shaofu
ZTE

Email: peng.shaofu@zte.com.cn

Wim Henderickx
Nokia

Email: wim.henderickx@nokia.com

Network Working Group
Internet-Draft
Updates: rfc1191, rfc4443, rfc7526,
 rfc8201 (if approved)
Intended status: Standards Track
Expires: September 25, 2021

F. Templin, Ed.
The Boeing Company
A. Whyman
MWA Ltd c/o Inmarsat Global Ltd
March 24, 2021

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

(<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	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
12.1.9. Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Message	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

13. Address Mapping - Multicast	63
14. Multilink Conceptual Sending Algorithm	63
14.1. Multiple OMNI Interfaces	64
14.2. MN<->AR Traffic Loop Prevention	64
15. Router Discovery and Prefix Registration	65
15.1. Router Discovery in IP Multihop and IPv4-Only Networks .	69
15.2. MS-Register and MS-Release List Processing	71
15.3. DHCPv6-based Prefix Registration	73
16. Secure Redirection	74
17. AR and MSE Resilience	74
18. Detecting and Responding to MSE Failures	75
19. Transition Considerations	75
20. OMNI Interfaces on Open Internetworks	76
21. Time-Varying MNPs	78
22. (H)HITs and Temporary ULAs	78
23. Address Selection	79
24. Error Messages	80
25. IANA Considerations	80
25.1. "IEEE 802 Numbers" Registry	80
25.2. "IPv6 Neighbor Discovery Option Formats" Registry . . .	80
25.3. "Ethernet Numbers" Registry	80
25.4. "ICMPv6 Code Fields: Type 2 - Packet Too Big" Registry .	81
25.5. "OMNI Option Sub-Type Values" (New Registry)	81
25.6. "OMNI Node Identification ID-Type Values" (New Registry)	82
25.7. "OMNI Option Sub-Type Extension Values" (New Registry) .	82
25.8. "OMNI RFC4380 UDP/IP Header Option" (New Registry) . . .	82
25.9. "OMNI RFC6081 UDP/IP Trailer Option" (New Registry) . .	83
25.10. Additional Considerations	83
26. Security Considerations	84
27. Implementation Status	85
28. Acknowledgements	85
29. References	86
29.1. Normative References	86
29.2. Informative References	88
Appendix A. Interface Attribute Preferences Bitmap Encoding . .	96
Appendix B. VDL Mode 2 Considerations	97
Appendix C. MN / AR Isolation Through L2 Address Mapping	98
Appendix D. Change Log	99
Authors' Addresses	101

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 (MNPs) are delegated. OMNI link administrators typically obtain MSPs from an Internet address registry, however private-use prefixes can alternatively be used subject to certain limitations (see: Section 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.

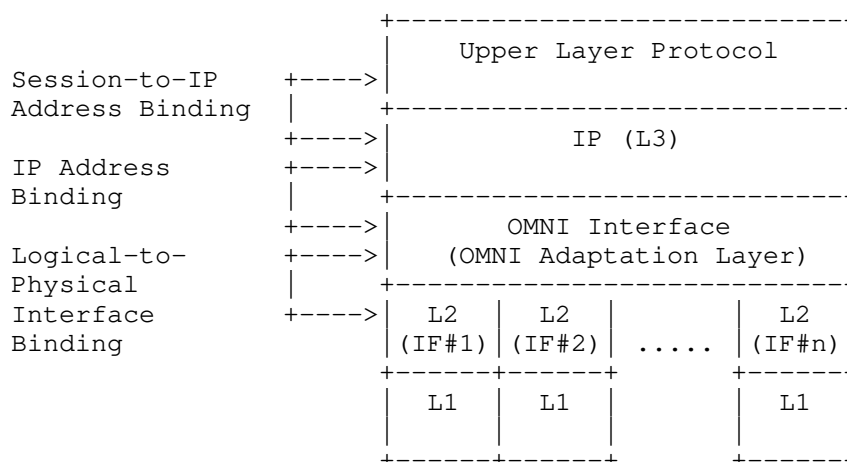


Figure 1: OMNI Interface Architectural Layering Model

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

- o INET interfaces connect to an INET either natively or through one or several IPv4 Network Address Translators (NATs). Native INET interfaces have global IP addresses that are reachable from any INET correspondent. NATed INET interfaces typically have private IP addresses and connect to a private network behind one or more NATs that provide INET access.
- o ANET interfaces connect to a protected ANET that is separated from the open INET by an AR acting as a proxy. The ANET interface may be either on the same L2 link segment as the AR, or separated from the AR by multiple IP hops.
- o VPned interfaces use security encapsulation over a *NET to a Virtual Private Network (VPN) gateway. Other than the link-layer

encapsulation format, VPNed interfaces behave the same as for Direct interfaces.

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

The OMNI 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.

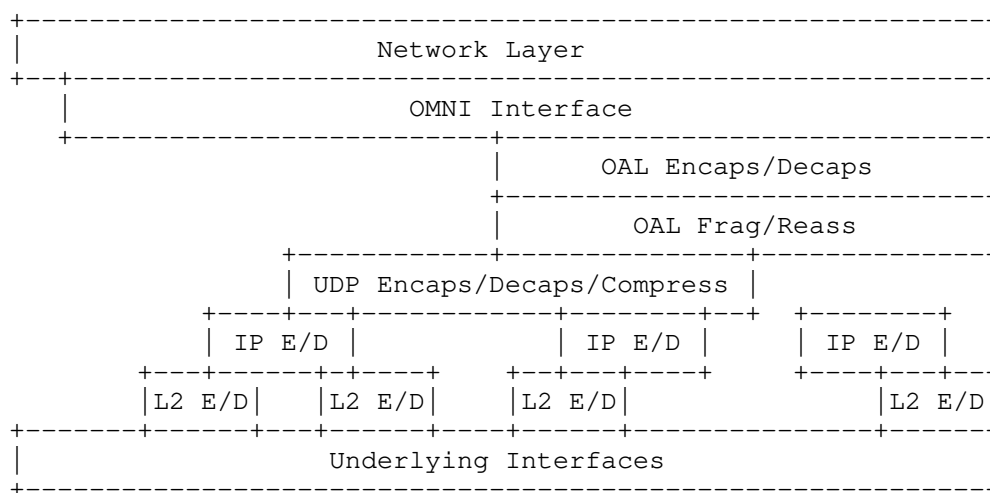


Figure 2: OMNI Interface Layering

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

- o 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.
- o 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.
- o 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.

- o as underlying interface properties change (e.g., link quality, cost, availability, etc.), any active interface can be used to update the profiles of multiple additional interfaces in a single message. This allows for timely adaptation and service continuity under dynamically changing conditions.
- o coordinating underlying interfaces in this way allows them to be represented in a unified MS profile with provisions for mobility and multilink operations.
- o exposing a single virtual interface abstraction to the IPv6 layer allows for multilink operation (including QoS based link selection, packet replication, load balancing, etc.) at L2 while still permitting L3 traffic shaping based on, e.g., DSCP, flow label, etc.
- o the OMNI interface allows inter-INET traversal when nodes located in different INETs need to communicate with one another. This mode of operation would not be possible via direct communications over the underlying interfaces themselves.
- o the 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.
- o the OAL applies per-packet identification values that allow for link-layer reliability and data origin authentication.
- o L3 sees the OMNI interface as a point of connection to the OMNI link; if there are multiple OMNI links (i.e., multiple MS's), L3 will see multiple OMNI interfaces.
- o Multiple independent OMNI interfaces can be used for increased fault tolerance through Safety-Based Multilink (SBM), with Performance-Based Multilink (PBM) applied within each interface.

Other opportunities are discussed in [RFC7847]. Note that even when the OMNI virtual interface is present, applications can still access underlying interfaces either through the network protocol stack using an Internet socket or directly using a raw socket. This allows for intra-network (or point-to-point) communications without invoking the OMNI interface and/or OAL. For example, when an IPv6 OMNI interface is configured over an underlying IPv4 interface, applications can still invoke IPv4 intra-network communications as long as the

communicating endpoints are not subject to mobility dynamics. However, the opportunities discussed above are not 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.

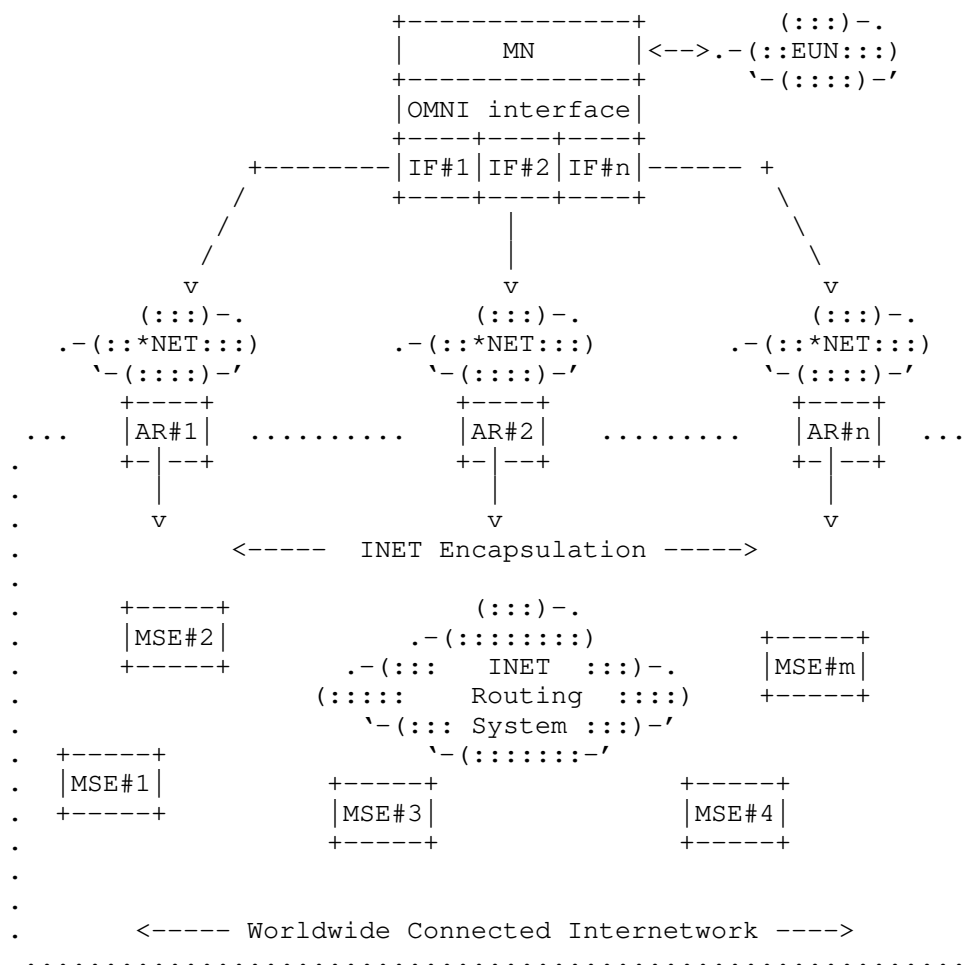


Figure 3: MN/MS Coordination via Multiple *NETs

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

in turn uses *NET encapsulation to forward carrier packets over the underlying interfaces (see: Section 6).

6. 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:

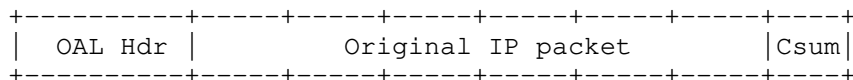


Figure 4: OAL Packet Before Fragmentation

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:

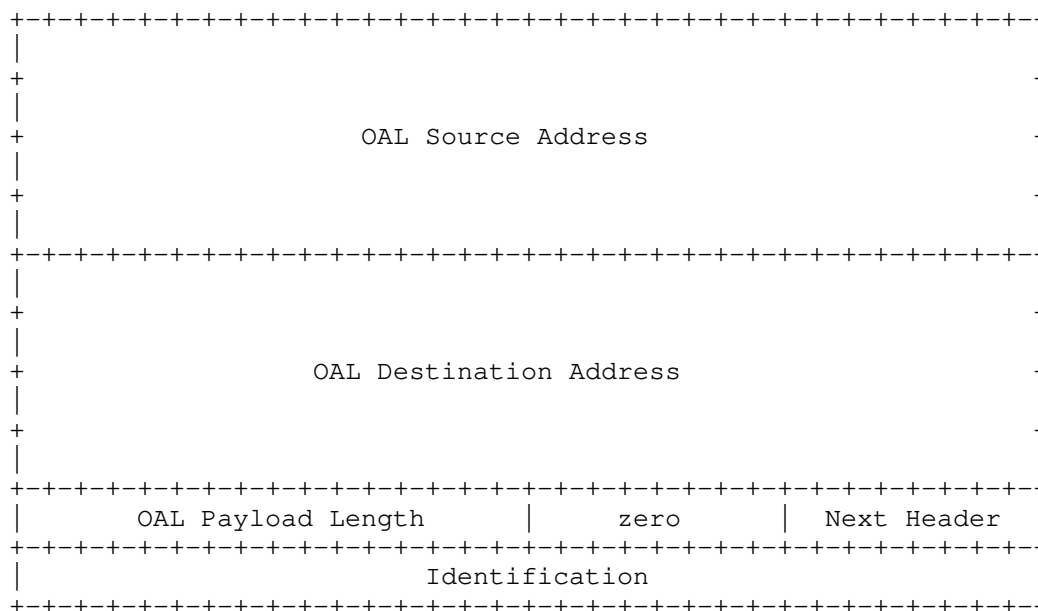


Figure 5: OAL IPv6 Pseudo-Header

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.

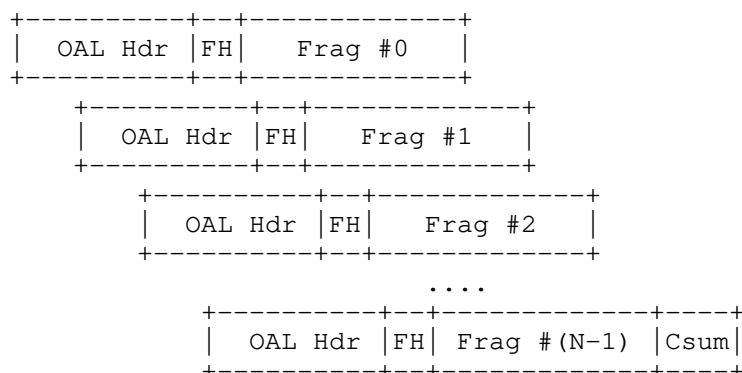
The OAL source therefore assumes a default minimum path MTU of 576 bytes at each *NET segment for the purpose of generating OAL fragments for *NET encapsulation and transmission as carrier packets. In the worst case, each successive *NET segment may re-encapsulate with either a 20 byte IPv4 or 40 byte 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.

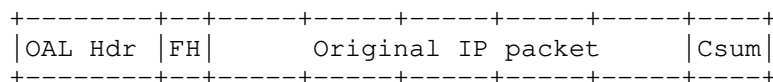
The OAL source can also actively probe individual OAL destinations to discover larger path MPS values using packetization layer probes per [RFC4821][RFC8899], but care must be taken to avoid setting static values for dynamically changing paths leading to black holes. The probe involves sending an OAL 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).

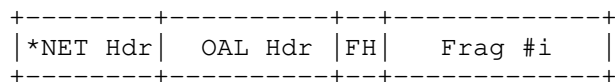
When the OAL source performs fragmentation, it SHOULD produce the minimum number of non-overlapping fragments under current MPS constraints, where each non-final fragment MUST be of equal length at least as large as the minimum MPS, while the final fragment MAY be of different length. The OAL source 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.



- a) OAL fragments after fragmentation
(FH = Fragment Header; Csum appears only in final fragment)



- b) An OAL atomic fragment with FH but no fragmentation.



- c) OAL carrier packet after *NET encapsulation

Figure 6: OAL Fragments and Carrier Packets

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:

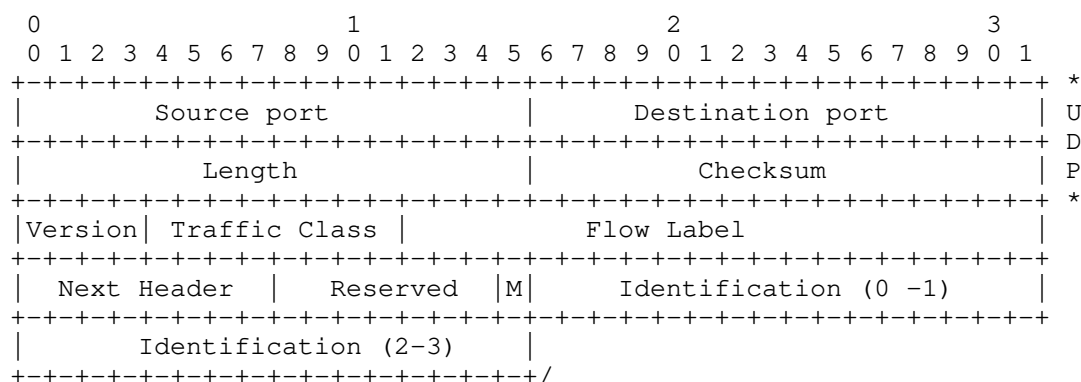


Figure 7: OMNI Compressed Header - Type 0 (OCH-0)

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:

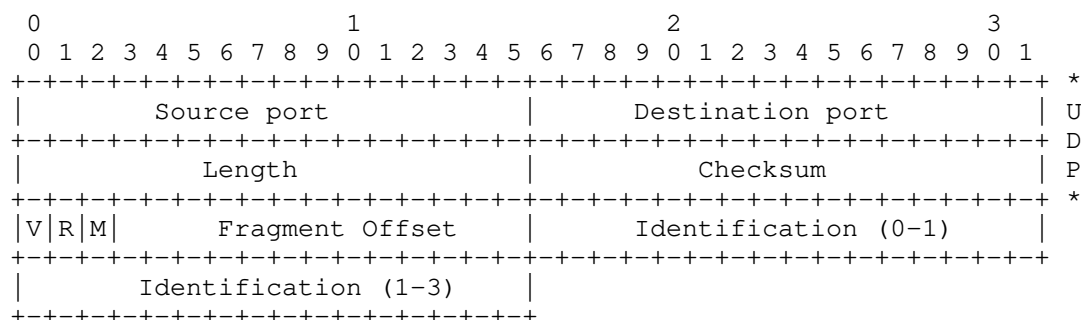


Figure 8: OMNI Compressed Header - Type 1 (OCH-1)

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:

- o 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.
- o 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.
- o OAL sources MUST NOT send OAL fragments that include any extension headers other than a single ORH and a single Fragment Header.
- o 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 hard Reassembly Limit, i.e., whether or not reassembly was needed.
- o OAL intermediate nodes SHOULD and OAL destinations MUST unconditionally drop any non-final OAL fragments containing a payload smaller than the minimum MPS.
- o 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:

[I-D.ietf-intarea-tunnels]) is also supported so that multiple original IP packets and/or control messages can be included within a single OAL "super-packet".

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:

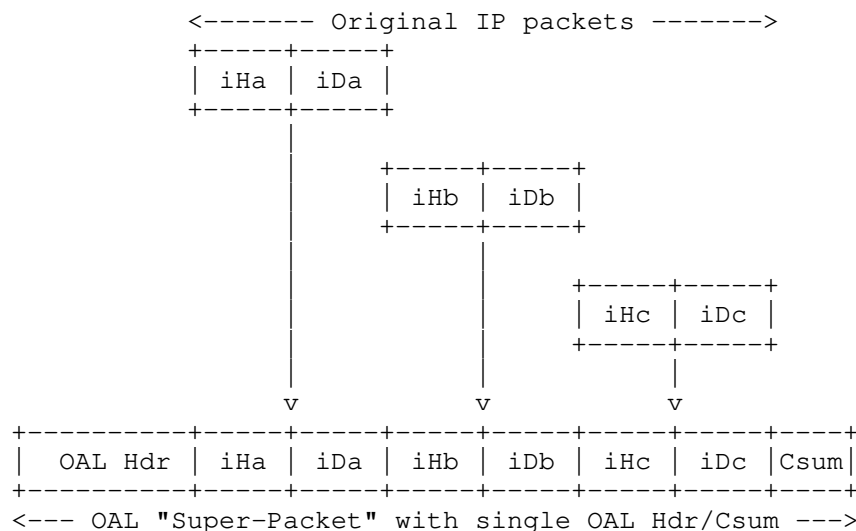


Figure 9: OAL Super-Packet Format

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:

- o 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.
- o IPv4-compatible MNP-LLAs are constructed as fe80::ffff:[IPv4], i.e., the interface identifier consists of 16 '0' bits, followed by 16 '1' bits, followed by a 32bit IPv4 address/prefix. The

prefix length for the LLA is determined by adding 96 to the MNP prefix length. For example, the IPv4-Compatible MN OMNI LLA for 192.0.2.0/24 is fe80::ffff:192.0.2.0/120 (also written as fe80::ffff:c000:0200/120).

- o ADM-LLAs are assigned to ARs and MSEs and MUST be managed for uniqueness. The lower 32 bits of the LLA includes a unique integer "MSID" value between 0x00000001 and 0xfeffffff, e.g., as in fe80::1, fe80::2, fe80::3, etc., fe80::feffffff. The ADM-LLA prefix length is determined by adding 96 to the MSID prefix length. For example, if the prefix length for MSID 0x10012001 is 16 then the ADM-LLA prefix length is set to 112 and the LLA is written as fe80::1001:2001/112. The "zero" address for each ADM-LLA prefix is the Subnet-Router anycast address for that prefix [RFC4291]; for example, the Subnet-Router anycast address for fe80::1001:2001/112 is simply fe80::1001:2000. The MSID range 0xff000000 through 0xffffffff is reserved for future use.

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 a an OMNI domain, and are derived from the IPv6 Unique Local Address prefix `fc00::/7` followed by the L bit set to 1 (i.e., as `fd00::/8`) followed by a 40-bit pseudo-random Global ID to produce the prefix `[ULA]::/48`, which is then followed by a 16-bit Subnet ID then finally followed by a 64 bit Interface ID as specified in Section 3 of [RFC4193]. 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 `0xfeff` in bits 48-63 of `[ULA]::/48`; the values `0xff00` through `0xffff` 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`:

- o the MNP-ULA corresponding to the MNP-LLA `fe80::2001:db8:1:2` with a 56-bit MNP length is derived by copying the lower 64 bits of the LLA into the lower 64 bits of the ULA as `[ULA]:1010:2001:db8:1:2/120` (where, the ULA prefix length becomes 64 plus the IPv6 MNP length).
- o the MNP-ULA corresponding to `fe80::ffff:192.0.2.0` with a 28-bit MNP length is derived by simply writing the LLA interface ID into the lower 64 bits as `[ULA]:1010:0:ffff:192.0.2.0/124` (where, the ULA prefix length is 64 plus 32 plus the IPv4 MNP length).
- o the ADM-ULA corresponding to `fe80::1000/112` is simply `[ULA]:1010::1000/112`.
- o the ADM-ULA corresponding to `fe80::/128` is simply `[ULA]:1010::/128`.
- o etc.

Each OMNI interface assigns the Anycast ADM-ULA specific to the OMNI link instance. For example, the OMNI interface connected to instance

3 assigns the Anycast address [ULA]:0003::/128. Routers that configure OMNI interfaces advertise the OMNI service prefix (e.g., [ULA]:0003::/64) into the local routing system so that applications can direct traffic according to SBM requirements.

The ULA presents an IPv6 address format that is routable within the OMNI routing system and can be used to convey link-scoped IPv6 ND messages across multiple hops using IPv6 encapsulation [RFC2473]. The OMNI link extends across one or more underling Internetworks to include all ARs and MSEs. All MNs are also considered to be connected to the OMNI link, however OAL encapsulation is omitted whenever possible to conserve bandwidth (see: Section 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:

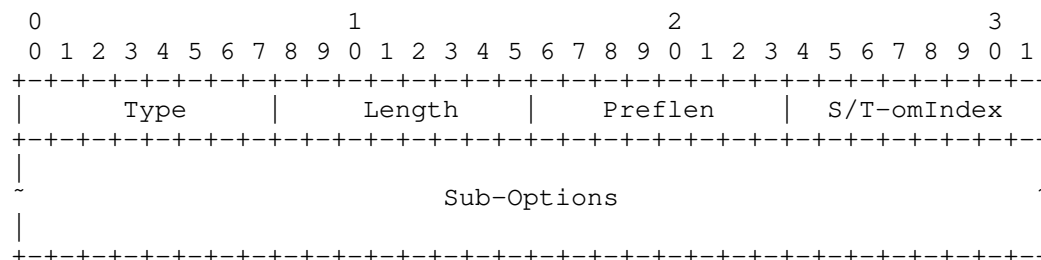


Figure 10: OMNI Option Format

In this format:

- o Type is set to TBD2.
- o 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.
- o 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.

- o 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.)
- o Sub-Options is a Variable-length field, of length such that the complete OMNI Option is an integer multiple of 8 octets long. Contains one or more Sub-Options, as described in Section 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:

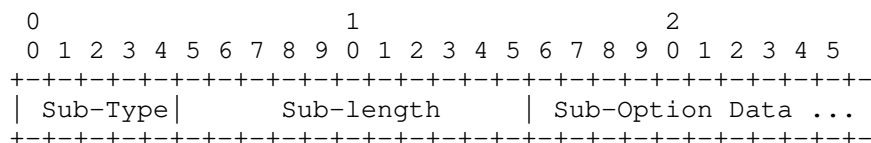


Figure 11: Sub-Option Format

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

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

Figure 12

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

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

12.1.1.1. Pad1

```

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

Figure 13: Pad1

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

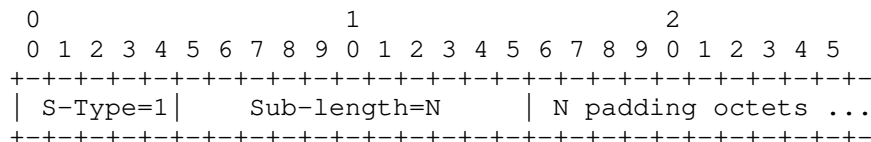


Figure 14: PadN

- o Sub-Type is set to 1. If multiple instances appear in OMNI options of the same message all are processed.
- o Sub-Length is set to N (from 0 to 2034) that encodes the number of padding octets that follow.
- o 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:

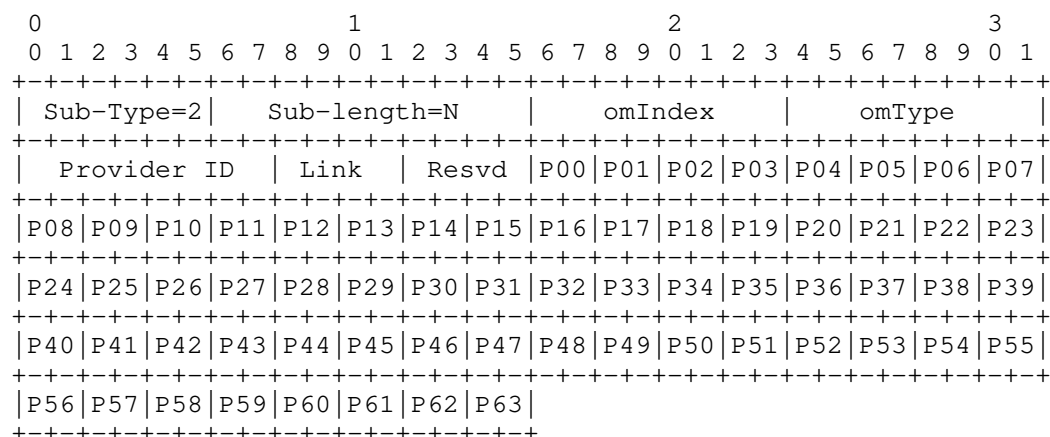


Figure 15: Interface Attributes (Type 1)

- o 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
- o Sub-Length is set to N (from 4 to 2034) that encodes the number of Sub-Option Data octets that follow.
- o omIndex is a 1-octet field containing a value from 0 to 255 identifying the underlying interface for which the attributes apply.
- o omType is a 1-octet field containing a value from 0 to 255 corresponding to the underlying interface identified by omIndex.
- o Provider ID is a 1-octet field containing a value from 0 to 255 corresponding to the underlying interface identified by omIndex.
- o 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").
- o Resvd is reserved for future use. Set to 0 on transmission and ignored on reception.
- o 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:

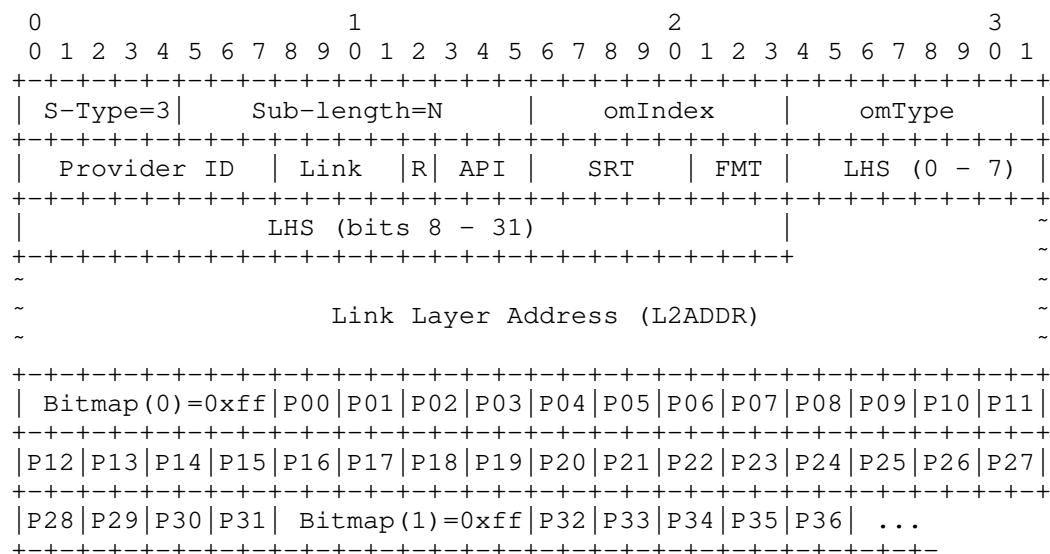


Figure 16: Interface Attributes (Type 2)

- o 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.

- o 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.
- o 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

block immediately follows L2ADDR; else, it immediately follows the API code.)

- + When a preferences block is present and the least significant bit (i.e., "Indexed") is set to 0, the block is encoded in "Simplex" form as shown in Figure 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-compliment "obfuscated" form per [RFC4380]. The OMNI interface forwarding algorithm uses FMT/L2ADDR to determine the encapsulation address for forwarding when SRT/LHS is located in the local OMNI link segment. Note that if the target is behind a NAT, L2ADDR will contain the mapped INET address stored in the NAT; otherwise, L2ADDR will contain the native INET information of the target itself.
- * When API indicates that "Preferences" are included, a preferences block appears as the remainder of the Sub-Option as a series of Bitmaps and P[*] values. In "Simplex" form, the index for each singleton Bitmap octet is inferred from its sequential position (i.e., 0, 1, 2, ...) as shown in Figure 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 ...
++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-

```

Figure 17

- * 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

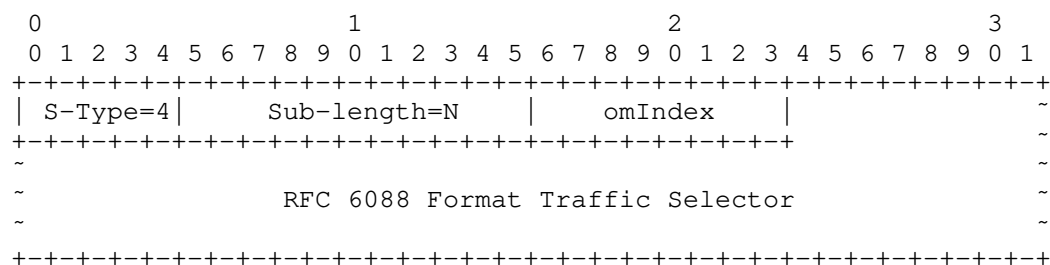


Figure 18: Traffic Selector

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

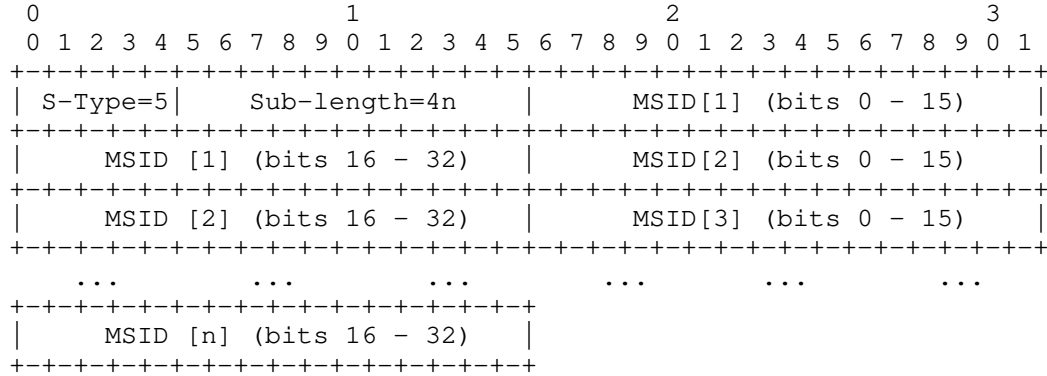


Figure 19: MS-Register Sub-option

- o 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.
- o 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.
- o A list of n 4 octet MSIDs is included in the following 4n octets. The Anycast MSID value '0' in an RS message MS-Register sub-option requests the recipient to return the MSID of a nearby MSE in a corresponding RA response.

12.1.7. MS-Release

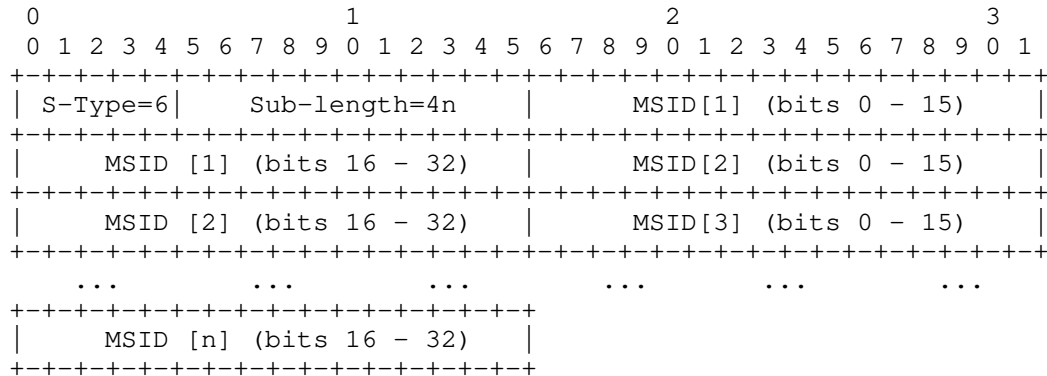


Figure 20: MS-Release Sub-option

- o 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.
- o 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.
- o A list of n 4 octet MSIDs is included in the following 4n octets. The Anycast MSID value '0' is ignored in MS-Release sub-options, i.e., only non-zero values are processed.

12.1.8. Geo Coordinates

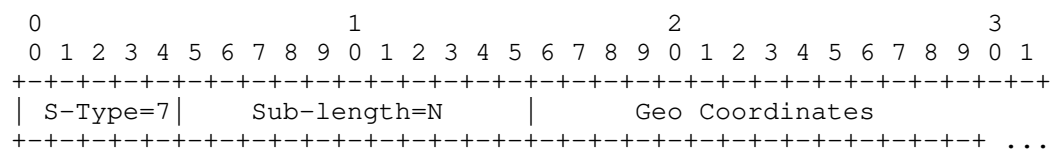


Figure 21: Geo Coordinates Sub-option

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

12.1.9. Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Message

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.

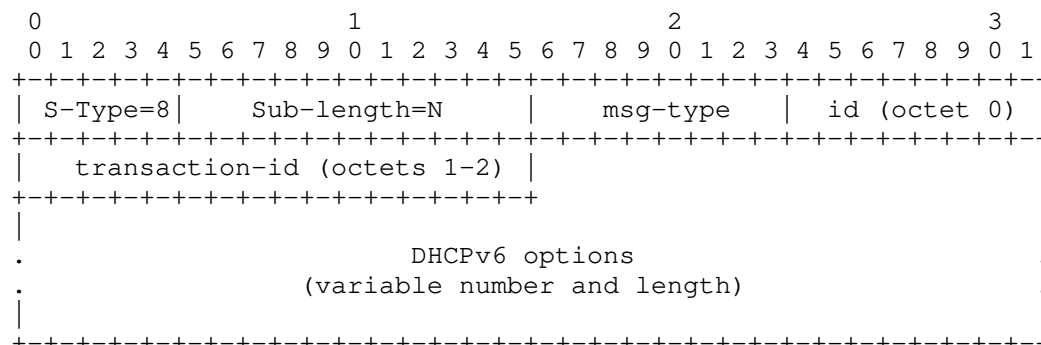


Figure 22: DHCPv6 Message Sub-option

- o 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.
- o 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.
- o 'msg-type' and 'transaction-id' are coded according to Section 8 of [RFC8415].
- o 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.

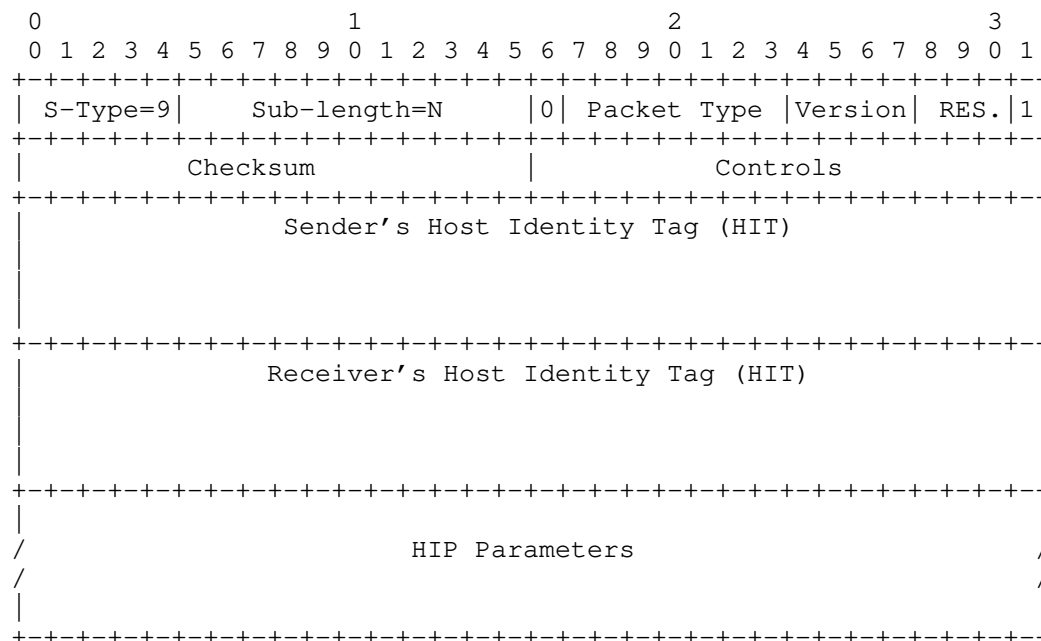


Figure 23: HIP Message Sub-option

- o 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.
- o 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.
- o 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.

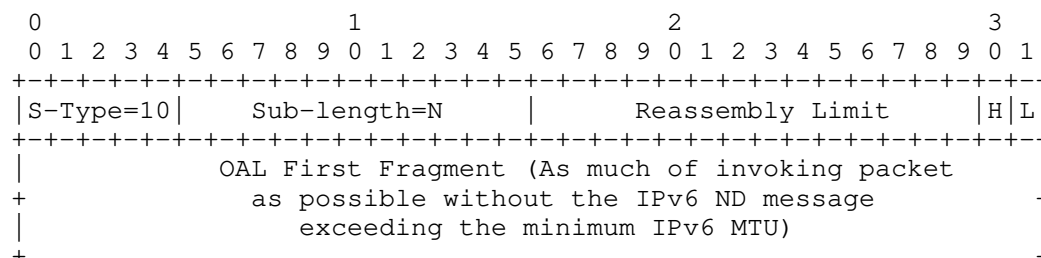


Figure 24: Reassembly Limit

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

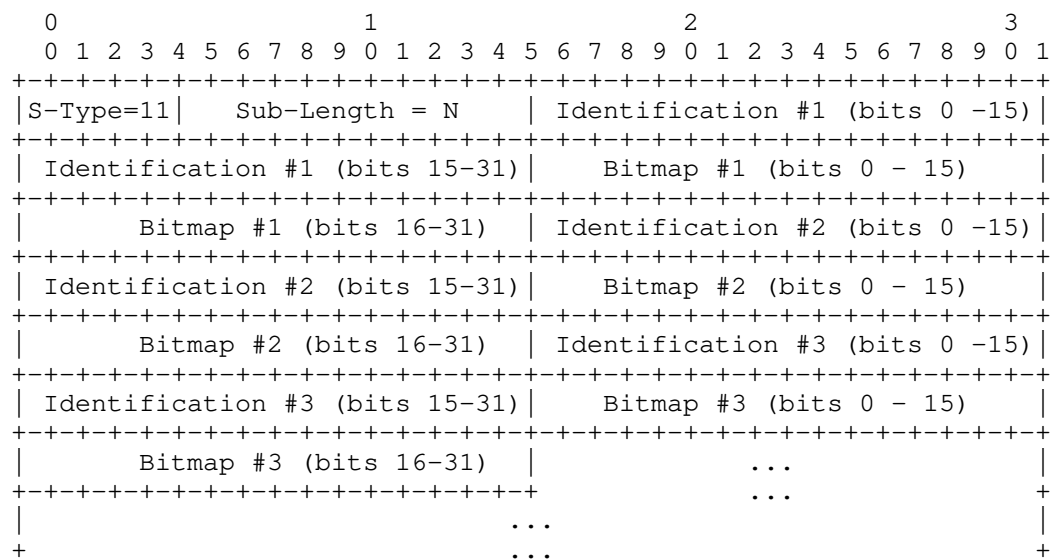


Figure 25: Fragmentation Report

- o Sub-Type is set to 11. If multiple instances appear in OMNI options of the same message all are processed.
- o 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.
- o 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.)

- o 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
    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|1|0|1|1|1|1|1|1|0|1|1|0|1|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

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|S-Type=12| Sub-length=N | ID-Type | ~
+-----+-----+-----+-----+-----+-----+
~ Node Identification Value (N-1 octets) ~
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 27: Node Identification

- o 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.)
- o 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.
- o 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.
- o 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:

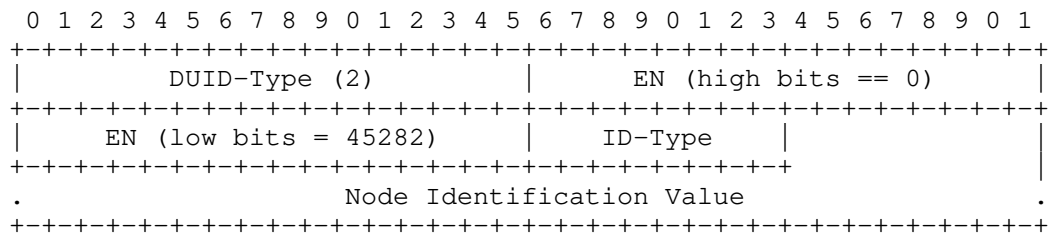


Figure 28: DUID-EN for OMNI Format

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:

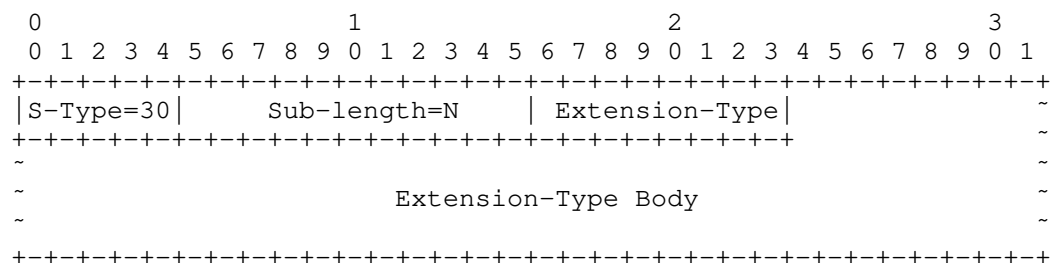


Figure 29: Sub-Type Extension

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

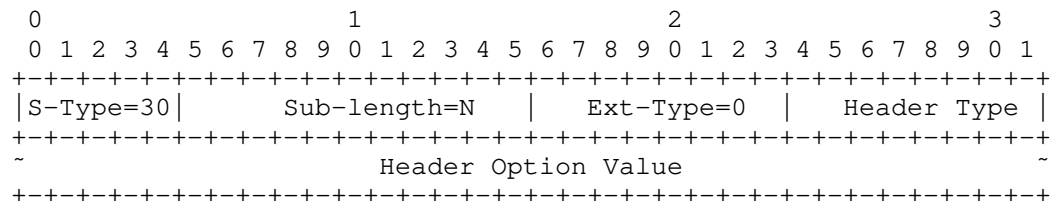


Figure 30: RFC4380 UDP/IP Header Option (Extension-Type 0)

- o Sub-Type is set to 30.
- o 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.
- o 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.
- o 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.
- o 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

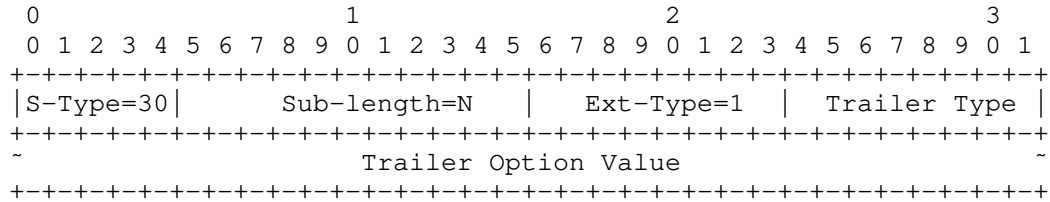


Figure 31: RFC6081 UDP/IP Trailer Option (Extension-Type 1)

- o Sub-Type is set to 30.
- o 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.
- o 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.
- o 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

MNs may connect to multiple independent OMNI links concurrently in support of SBM. Each OMNI interface is distinguished by its Anycast ULA (e.g., [ULA]:0002::, [ULA]:1000::, [ULA]:7345::, etc.). The MN configures a separate OMNI interface for each link so that multiple interfaces (e.g., omni0, omni1, omni2, etc.) are exposed to the IPv6 layer. A different Anycast ULA is assigned to each interface, and the MN injects the service prefixes for the OMNI link instances into the EUN routing system.

Applications in EUNs can use Segment Routing to select the desired OMNI interface based on SBM considerations. The Anycast ULA is written into 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:

- o if the IP destination address belongs to a neighbor on the same OMNI interface, and if the link-layer source address is the same as one of the neighbor's link-layer addresses, drop the packet.

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:

- o PIOs with (A; L=0) that include MSPs for the link [RFC8028].
- o RIOs [RFC4191] with more-specific routes.

- o an MTU option that specifies the maximum acceptable packet size for this underlying interface.

The 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:

- o 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.
- o 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).

- o 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.
- o 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.
- o 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 repurpose 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:

- o For each list, retain the first MAX_MSID values in the list and discard any additional MSIDs (i.e., even if there are duplicates within a list).
- o Next, for each MSID in the MS-Register list, remove all matching MSIDs from the MS-Release list.
- o Next, proceed 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

MNs 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

local area. 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:

Addresses	Usage	Reference
-----	-----	-----
00-52-14	Overlay Multilink Network (OMNI) Interface	[RFCXXXX]

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:

Code ---	Name -----	Reference -----
0	PTB Hard Error	[RFC4443]
1	PTB Soft Error (loss)	[RFCXXXX]
2	PTB Soft Error (no loss)	[RFCXXXX]

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]):

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

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]):

Value	Sub-Type name	Reference
-----	-----	-----
0	UUID	[RFCXXXX]
1	HIT	[RFCXXXX]
2	HHIT	[RFCXXXX]
3	Network Access Identifier	[RFCXXXX]
4	FQDN	[RFCXXXX]
5-252	Unassigned	[RFCXXXX]
253-254	Reserved for Experimentation	[RFCXXXX]
255	Reserved by IANA	[RFCXXXX]

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]):

Value	Sub-Type name	Reference
-----	-----	-----
0	RFC4380 UDP/IP Header Option	[RFCXXXX]
1	RFC6081 UDP/IP Trailer Option	[RFCXXXX]
2-252	Unassigned	
253-254	Reserved for Experimentation	[RFCXXXX]
255	Reserved by IANA	[RFCXXXX]

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]):

Value	Sub-Type name	Reference
-----	-----	-----
0	Origin Indication (IPv4)	[RFC4380]
1	Authentication Encapsulation	[RFC4380]
2	Origin Indication (IPv6)	[RFCXXXX]
3-252	Unassigned	
253-254	Reserved for Experimentation	[RFCXXXX]
255	Reserved by IANA	[RFCXXXX]

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]):

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

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-D.templin-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, Aloke Roy, Greg Saccone, Robert Segers, Michal Skorepa, Michel Solery, Stephane Tamalet, Fred Templin, Jean-Marc Vacher, Bela Varkonyi, Tony Whyman, Fryderyk Wrobel and Dongsong Zeng.

The following individuals are acknowledged for their useful comments: 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, Vijayasaratthy 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 Juzsaczak 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

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, DOI 10.17487/RFC4727, November 2006, <<https://www.rfc-editor.org/info/rfc4727>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC6088] Tsirtsis, G., Giarreta, G., Soliman, H., and N. Montavont, "Traffic Selectors for Flow Bindings", RFC 6088, DOI 10.17487/RFC6088, January 2011, <<https://www.rfc-editor.org/info/rfc6088>>.

- [RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <<https://www.rfc-editor.org/info/rfc7401>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<https://www.rfc-editor.org/info/rfc8028>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.
- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.

29.2. Informative References

- [ATN] Maiolla, V., "The OMNI Interface - An IPv6 Air/Ground Interface for Civil Aviation, IETF Liaison Statement #1676, <https://datatracker.ietf.org/liaison/1676/>", March 2020.
- [ATN-IPS] WG-I, ICAO., "ICAO Document 9896 (Manual on the Aeronautical Telecommunication Network (ATN) using Internet Protocol Suite (IPS) Standards and Protocol), Draft Edition 3 (work-in-progress)", December 2020.
- [CKSUM] Stone, J., Greenwald, M., Partridge, C., and J. Hughes, "Performance of Checksums and CRC's Over Real Data, IEEE/ACM Transactions on Networking, Vol. 6, No. 5", October 1998.

- [CRC] Jain, R., "Error Characteristics of Fiber Distributed Data Interface (FDDI), IEEE Transactions on Communications", August 1990.
- [I-D.ietf-drip-rid] Moskowitz, R., Card, S., Wiethuechter, A., and A. Gurtov, "UAS Remote ID", draft-ietf-drip-rid-06 (work in progress), December 2020.
- [I-D.ietf-intarea-tunnels] Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-10 (work in progress), September 2019.
- [I-D.ietf-ipwave-vehicular-networking] Jeong, J., "IPv6 Wireless Access in Vehicular Environments (IPWAVE): Problem Statement and Use Cases", draft-ietf-ipwave-vehicular-networking-19 (work in progress), July 2020.
- [I-D.ietf-tsvwg-udp-options] Touch, J., "Transport Options for UDP", draft-ietf-tsvwg-udp-options-09 (work in progress), November 2020.
- [I-D.templin-6man-dhcpv6-ndopt] Templin, F., "A Unified Stateful/Stateless Configuration Service for IPv6", draft-templin-6man-dhcpv6-ndopt-11 (work in progress), January 2021.
- [I-D.templin-6man-lla-type] Templin, F., "The IPv6 Link-Local Address Type Field", draft-templin-6man-lla-type-02 (work in progress), November 2020.
- [I-D.templin-intarea-6706bis] Templin, F., "Asymmetric Extended Route Optimization (AERO)", draft-templin-intarea-6706bis-87 (work in progress), January 2021.
- [IPV4-GUA] Postel, J., "IPv4 Address Space Registry, <https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml>", December 2020.

[IPV6-GUA]

Postel, J., "IPv6 Global Unicast Address Assignments, <https://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml>", December 2020.

- [RFC0905] "ISO Transport Protocol specification ISO DP 8073", RFC 905, DOI 10.17487/RFC0905, April 1984, <<https://www.rfc-editor.org/info/rfc905>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<https://www.rfc-editor.org/info/rfc1122>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC1256] Deering, S., Ed., "ICMP Router Discovery Messages", RFC 1256, DOI 10.17487/RFC1256, September 1991, <<https://www.rfc-editor.org/info/rfc1256>>.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC2225] Laubach, M. and J. Halpern, "Classical IP and ARP over ATM", RFC 2225, DOI 10.17487/RFC2225, April 1998, <<https://www.rfc-editor.org/info/rfc2225>>.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <<https://www.rfc-editor.org/info/rfc2328>>.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <<https://www.rfc-editor.org/info/rfc2464>>.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.

- [RFC2492] Armitage, G., Schulter, P., and M. Jork, "IPv6 over ATM Networks", RFC 2492, DOI 10.17487/RFC2492, January 1999, <<https://www.rfc-editor.org/info/rfc2492>>.
- [RFC2529] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529, DOI 10.17487/RFC2529, March 1999, <<https://www.rfc-editor.org/info/rfc2529>>.
- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <<https://www.rfc-editor.org/info/rfc2863>>.
- [RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, DOI 10.17487/RFC2923, September 2000, <<https://www.rfc-editor.org/info/rfc2923>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<https://www.rfc-editor.org/info/rfc2983>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3330] IANA, "Special-Use IPv4 Addresses", RFC 3330, DOI 10.17487/RFC3330, September 2002, <<https://www.rfc-editor.org/info/rfc3330>>.
- [RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, DOI 10.17487/RFC3692, January 2004, <<https://www.rfc-editor.org/info/rfc3692>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <<https://www.rfc-editor.org/info/rfc3819>>.

- [RFC3879] Huitema, C. and B. Carpenter, "Deprecating Site Local Addresses", RFC 3879, DOI 10.17487/RFC3879, September 2004, <<https://www.rfc-editor.org/info/rfc3879>>.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <<https://www.rfc-editor.org/info/rfc4122>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <<https://www.rfc-editor.org/info/rfc4380>>.
- [RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <<https://www.rfc-editor.org/info/rfc4389>>.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, DOI 10.17487/RFC4429, April 2006, <<https://www.rfc-editor.org/info/rfc4429>>.
- [RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <<https://www.rfc-editor.org/info/rfc4541>>.
- [RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, DOI 10.17487/RFC4605, August 2006, <<https://www.rfc-editor.org/info/rfc4605>>.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <<https://www.rfc-editor.org/info/rfc4821>>.

- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <<https://www.rfc-editor.org/info/rfc4963>>.
- [RFC5175] Haberman, B., Ed. and R. Hinden, "IPv6 Router Advertisement Flags Option", RFC 5175, DOI 10.17487/RFC5175, March 2008, <<https://www.rfc-editor.org/info/rfc5175>>.
- [RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, DOI 10.17487/RFC5213, August 2008, <<https://www.rfc-editor.org/info/rfc5213>>.
- [RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, DOI 10.17487/RFC5214, March 2008, <<https://www.rfc-editor.org/info/rfc5214>>.
- [RFC5558] Templin, F., Ed., "Virtual Enterprise Traversal (VET)", RFC 5558, DOI 10.17487/RFC5558, February 2010, <<https://www.rfc-editor.org/info/rfc5558>>.
- [RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <<https://www.rfc-editor.org/info/rfc5798>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC6081] Thaler, D., "Teredo Extensions", RFC 6081, DOI 10.17487/RFC6081, January 2011, <<https://www.rfc-editor.org/info/rfc6081>>.
- [RFC6221] Miles, D., Ed., Ooghe, S., Dec, W., Krishnan, S., and A. Kavanagh, "Lightweight DHCPv6 Relay Agent", RFC 6221, DOI 10.17487/RFC6221, May 2011, <<https://www.rfc-editor.org/info/rfc6221>>.
- [RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, DOI 10.17487/RFC6355, August 2011, <<https://www.rfc-editor.org/info/rfc6355>>.

- [RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <<https://www.rfc-editor.org/info/rfc6438>>.
- [RFC6543] Gundavelli, S., "Reserved IPv6 Interface Identifier for Proxy Mobile IPv6", RFC 6543, DOI 10.17487/RFC6543, May 2012, <<https://www.rfc-editor.org/info/rfc6543>>.
- [RFC6706] Templin, F., Ed., "Asymmetric Extended Route Optimization (AERO)", RFC 6706, DOI 10.17487/RFC6706, August 2012, <<https://www.rfc-editor.org/info/rfc6706>>.
- [RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <<https://www.rfc-editor.org/info/rfc6935>>.
- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <<https://www.rfc-editor.org/info/rfc6936>>.
- [RFC6980] Gont, F., "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery", RFC 6980, DOI 10.17487/RFC6980, August 2013, <<https://www.rfc-editor.org/info/rfc6980>>.
- [RFC7042] Eastlake 3rd, D. and J. Abley, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", BCP 141, RFC 7042, DOI 10.17487/RFC7042, October 2013, <<https://www.rfc-editor.org/info/rfc7042>>.
- [RFC7084] Singh, H., Beebe, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, DOI 10.17487/RFC7084, November 2013, <<https://www.rfc-editor.org/info/rfc7084>>.
- [RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <<https://www.rfc-editor.org/info/rfc7421>>.
- [RFC7526] Troan, O. and B. Carpenter, Ed., "Deprecating the Anycast Prefix for 6to4 Relay Routers", BCP 196, RFC 7526, DOI 10.17487/RFC7526, May 2015, <<https://www.rfc-editor.org/info/rfc7526>>.

- [RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <<https://www.rfc-editor.org/info/rfc7542>>.
- [RFC7739] Gont, F., "Security Implications of Predictable Fragment Identification Values", RFC 7739, DOI 10.17487/RFC7739, February 2016, <<https://www.rfc-editor.org/info/rfc7739>>.
- [RFC7847] Melia, T., Ed. and S. Gundavelli, Ed., "Logical-Interface Support for IP Hosts with Multi-Access Support", RFC 7847, DOI 10.17487/RFC7847, May 2016, <<https://www.rfc-editor.org/info/rfc7847>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8402] Filts, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8754] Filts, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.
- [RFC8892] Thaler, D. and D. Romascanu, "Guidelines and Registration Procedures for Interface Types and Tunnel Types", RFC 8892, DOI 10.17487/RFC8892, August 2020, <<https://www.rfc-editor.org/info/rfc8892>>.
- [RFC8899] Fairhurst, G., Jones, T., Tuexen, M., Ruengeler, I., and T. Voelker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <<https://www.rfc-editor.org/info/rfc8899>>.
- [RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <<https://www.rfc-editor.org/info/rfc8900>>.
- [RFC8981] Gont, F., Krishnan, S., Narten, T., and R. Draves, "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6", RFC 8981, DOI 10.17487/RFC8981, February 2021, <<https://www.rfc-editor.org/info/rfc8981>>.

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:

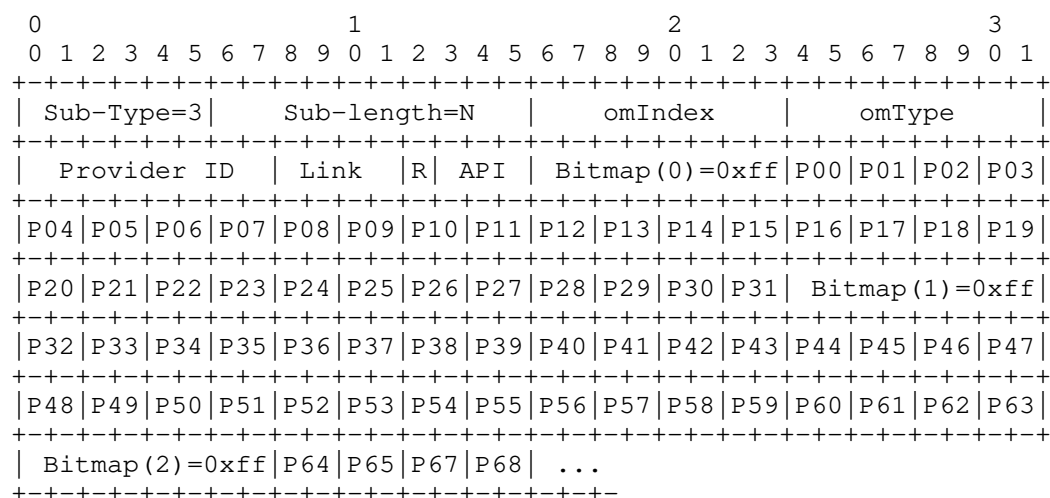


Figure 39: Example 1: Dense Simplex Encoding

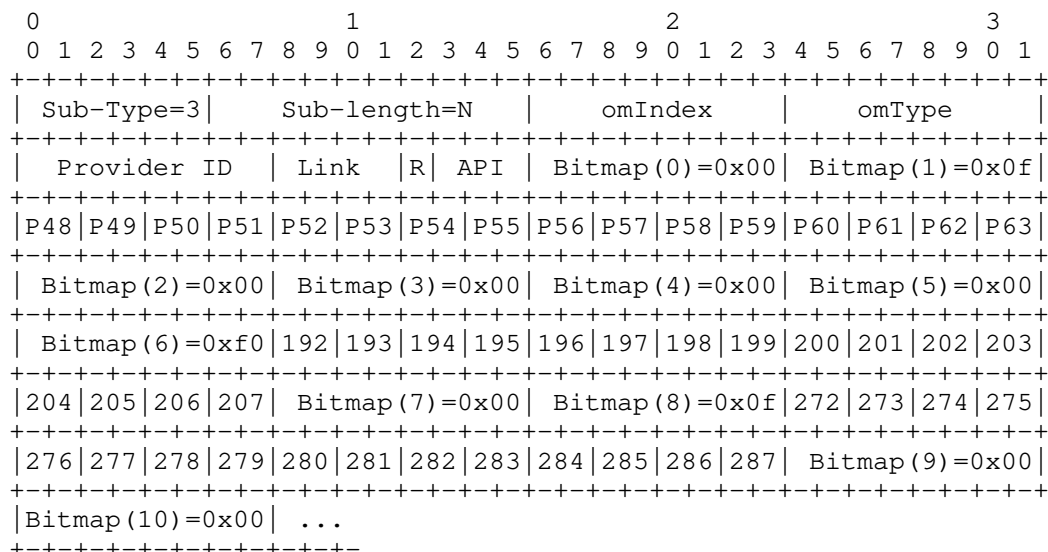


Figure 40: Example 2: Sparse Simplex Encoding

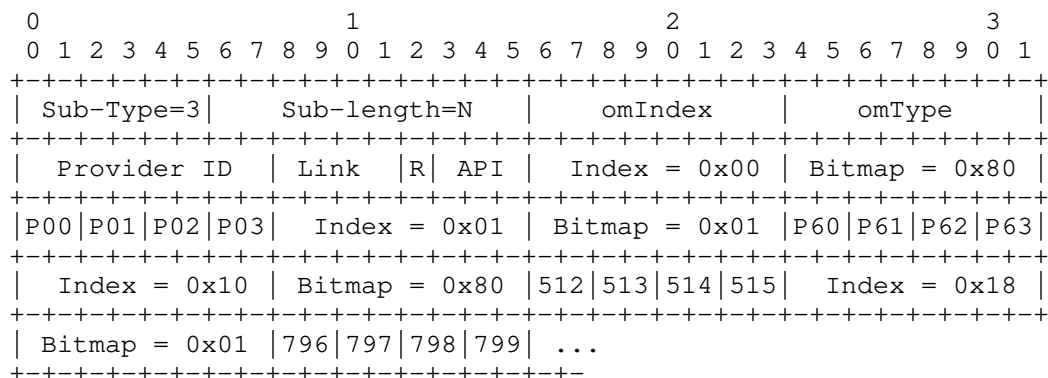


Figure 41: Example 3: Indexed Encoding

Appendix B. VDL Mode 2 Considerations

ICAO Doc 9776 is the "Technical Manual for VHF Data Link Mode 2" (VDLM2) that specifies an essential radio frequency data link service for aircraft and ground stations in worldwide civil aviation air traffic management. The VDLM2 link type is "multicast capable" [RFC4861], but with considerable differences from common multicast links such as Ethernet and IEEE 802.11.

First, the VDLM2 link data rate is only 31.5Kbps - multiple orders of magnitude less than most modern wireless networking gear. Second, due to the low available link bandwidth only VDLM2 ground stations (i.e., and not aircraft) are permitted to send broadcasts, and even so only as compact layer 2 "beacons". Third, aircraft employ the services of ground stations by performing unicast RS/RA exchanges upon receipt of beacons instead of listening for multicast RA messages and/or sending multicast RS messages.

This beacon-oriented unicast RS/RA approach is necessary to conserve the already-scarce available link bandwidth. Moreover, since the numbers of beaconing ground stations operating within a given spatial range must be kept as sparse as possible, it would not be feasible to have different classes of ground stations within the same region observing different protocols. It is therefore highly desirable that all ground stations observe a common language of RS/RA as specified in this document.

Note that links of this nature may benefit from compression techniques that reduce the bandwidth necessary for conveying the same amount of data. The IETF lpwan working group is considering possible alternatives: [<https://datatracker.ietf.org/wg/lpwan/documents>].

Appendix C. MN / AR Isolation Through L2 Address Mapping

Per [RFC4861], IPv6 ND messages may be sent to either a multicast or unicast link-scoped IPv6 destination address. However, IPv6 ND messaging should be coordinated between the MN and AR only without invoking other nodes on the *NET. This implies that MN / AR control messaging should be isolated and not overheard by other nodes on the link.

To support MN / AR isolation on some *NET links, ARs can maintain an OMNI-specific unicast L2 address ("MSADDR"). For Ethernet-compatible *NETs, this specification reserves one Ethernet unicast address 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:

- o Major clarifications on aspects such as "hard/soft" PTB error messages
- o Made generic so that either IP protocol version (IPv4 or IPv6) can be used in the data plane.

Differences from draft-templin-6man-omni-interface-31 to draft-templin-6man-omni-interface-32:

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

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

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

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

- o Updates based on implementation experience.

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

- o Further clarification on "aggregate" RA messages.
- o Expanded Security Considerations to discuss expectations for security in the Mobility Service.

Differences from draft-templin-6man-omni-interface-20 to draft-templin-6man-omni-interface-21:

- o Safety-Based Multilink (SBM) and Performance-Based Multilink (PBM).

Differences from draft-templin-6man-omni-interface-18 to draft-templin-6man-omni-interface-19:

- o SEND/CGA.

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

- o Teredo

Differences from draft-templin-6man-omni-interface-14 to draft-templin-6man-omni-interface-15:

- o Prefix length discussions removed.

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

- o Teredo

Differences from draft-templin-6man-omni-interface-11 to draft-templin-6man-omni-interface-12:

- o Major simplifications and clarifications on MTU and fragmentation.
- o Document now updates RFC4443 and RFC8201.

Differences from draft-templin-6man-omni-interface-10 to draft-templin-6man-omni-interface-11:

- o Removed /64 assumption, resulting in new OMNI address format.

Differences from draft-templin-6man-omni-interface-07 to draft-templin-6man-omni-interface-08:

- o OMNI MNs in the open Internet

Differences from draft-templin-6man-omni-interface-06 to draft-templin-6man-omni-interface-07:

- o Brought back L2 MSADDR mapping text for MN / AR isolation based on L2 addressing.
- o Expanded "Transition Considerations".

Differences from draft-templin-6man-omni-interface-05 to draft-templin-6man-omni-interface-06:

- o 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:

- o 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:

- o 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:

- o Removed "Primary" flag and supporting text.
- o 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:

- o "All-MSEs" OMNI LLA defined. Also reserved fe80::ff00:0000/104 for future use (most likely as "pseudo-multicast").
- o 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):

- o 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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2021

F. Templin, Ed.
The Boeing Company
A. Whyman
MWA Ltd c/o Inmarsat Global Ltd
February 12, 2021

IPv6 Neighbor Discovery Overlay Multilink Network Interface (OMNI)
Option
draft-templin-6man-omni-option-09

Abstract

This document defines a new IPv6 Neighbor Discovery (ND) option termed the "Overlay Multilink Network Interface (OMNI) Option". The OMNI option may appear in any IPv6 ND message type; it is processed by interface types that recognize the option and ignored by all other interface types. The option supports functions such as prefix registration and multilink coordination, and is extensible to support additional functions in the future.

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 August 16, 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
2. Terminology	2
3. The Overlay Multilink Network Interface (OMNI) IPv6 ND Option	3
3.1. Sub-Options	4
3.1.1. Pad1	5
3.1.2. PadN	6
3.1.3. Interface Attributes (Type 1)	6
3.1.4. Sub-Type Extension	8
4. IANA Considerations	9
5. Security Considerations	9
6. Acknowledgements	9
7. References	9
7.1. Normative References	9
7.2. Informative References	10
Authors' Addresses	11

1. Introduction

This document defines a new IPv6 Neighbor Discovery (ND) option termed the "Overlay Multilink Network Interface (OMNI) Option". The OMNI option may appear in any IPv6 ND message type; it is processed by interface types that recognize the option and ignored by all other interface types. The option supports functions such as prefix registration and multilink coordination for interface types such as the OMNI interface [I-D.templin-6man-omni-interface], and is extensible to support additional functions in the future.

The following sections discuss the OMNI option format and contents. Use cases appear in IPv6 over specific link layer documents such as [I-D.templin-6man-omni-interface], where the International Civil Aviation Organization (ICAO) has expressed interest in the option in support of their Document 9896 [ATN][ATN-IPS]. An IPv6 ND option Type number assignment is requested in the IANA Considerations section.

2. Terminology

The terminology in the normative references applies. The term "underlying interface" refers to one of potentially multiple Layer-2 interfaces over which a Layer-3 (virtual) interface is configured.

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

3. The Overlay Multilink Network Interface (OMNI) IPv6 ND Option

An Overlay Multilink Network Interface (OMNI) IPv6 ND option is defined. The option (known as the "OMNI option") is formatted as shown in Figure 1:

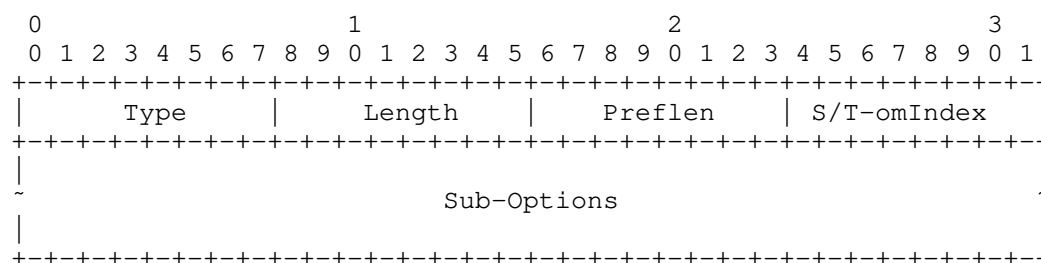


Figure 1: OMNI Option Format

In this format:

- o Type is set to TBD1.
- o Length is set to the number of 8 octet blocks in the option.
- o Preflen is an 8 bit field that determines the length of prefix associated with an IPv6 address of the IPv6 ND message. Values 0 through 128 specify a valid prefix length (all other values are invalid). For IPv6 ND messages sent from the source to a target node, Preflen applies to the IPv6 source address and provides the length that the source is requesting or asserting. For IPv6 ND messages replies from the target to the original source, Preflen applies to the IPv6 destination address and indicates the length that the target is granting.
- o S/T-omIndex is a 1-octet field that encodes a value between 0 and 255 identifying the source or target underlying interface for the IPv6 ND message. For RS and NS messages S/T-omIndex refers to the "Source" underlying interface over which the message is sent, while for RA and NA messages S/T-omIndex refers to the "Target" underlying interface that will receive the message.

- o Sub-Options is a Variable-length field, of length such that the complete OMNI Option is an integer multiple of 8 octets long. Contains one or more Sub-Options, as described in Section 3.1.

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 occur.

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.

3.1. Sub-Options

The OMNI option includes zero or more Sub-Options. Each consecutive Sub-Option is concatenated immediately after its predecessor. All Sub-Options except Pad1 (see below) are in type-length-value (TLV) encoded in the following format:

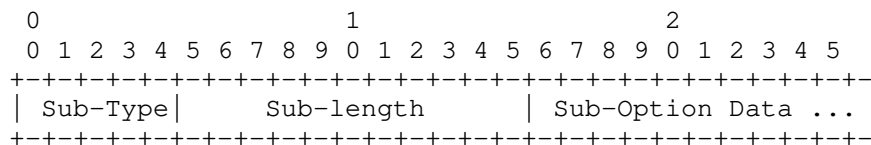


Figure 2: Sub-Option Format

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

Option Name	Sub-Type
Pad1	0
PadN	1
Interface Attributes (Type 1)	2
Sub-Type Extension	30

Figure 3

Sub-Types 3-29 are available for future assignment for major protocol functions. Sub-Type 31 is reserved by IANA.

- o Sub-Length is an 11-bit field that encodes the length of the Sub-Option Data (i.e., ranging from 0 to 2034 octets).
- o 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 character strings, 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.

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 the running total for that OMNI option to exceed the length coded in the option header, the interface accepts any Sub-Options already processed and ignores the remainder of that OMNI 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 (note that other documents that are active at the time of this writing will define additional Sub-Option types in the near future):

3.1.1. Pad1

```

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

```

Figure 4: Pad1

- o Sub-Type is set to 0. If multiple instances appear in OMNI options of the same message all are processed.
- o Sub-Type is followed by three 'x' bits, set randomly on transmission and ignored on receipt. Pad1 therefore consists of a whole single octet with the most significant 5 bits set to 0, and with no Sub-Length or Sub-Option Data fields following.

3.1.2. 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 5: PadN

- o Sub-Type is set to 1. If multiple instances appear in OMNI options of the same message all are processed.
- o Sub-Length is set to N (from 0 to 2047) being the number of padding octets that follow.
- o Sub-Option Data consists of N padding octets that are typically zero-valued (any non-zero values that may appear in the padding octets are not to be interpreted in any way other than as simple padding).

3.1.3. Interface Attributes (Type 1)

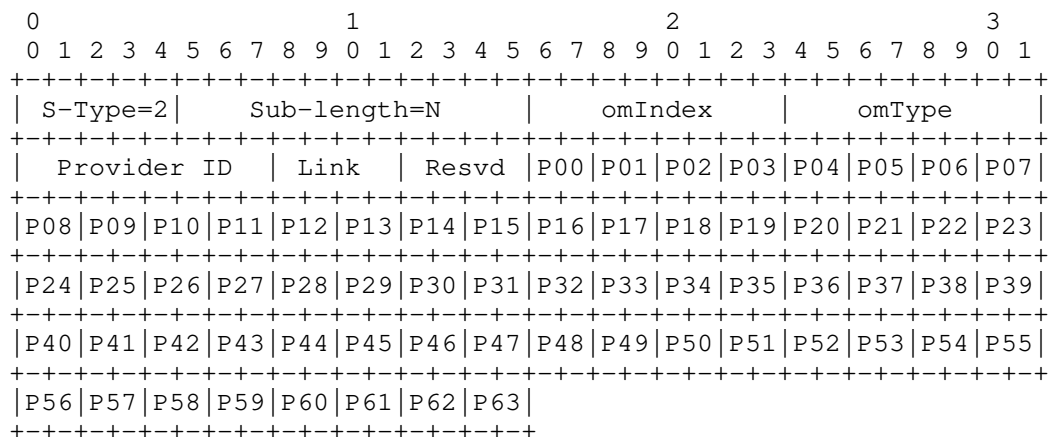


Figure 6: Interface Attributes (Type 1)

- o Sub-Type is set to 2. 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.
- o Sub-Length is set to N (from 1 to 2047) that encodes the number of Sub-Option Data octets that follow.
- o omIndex is a 1-octet field containing a value from 0 to 255 identifying the underlying interface for which the interface attributes apply.
- o omType is a 1-octet field containing a value from 0 to 255 corresponding to the underlying interface identified by omIndex.
- o Provider ID is a 1-octet field containing a value from 0 to 255 corresponding to the underlying interface identified by omIndex.
- o 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").
- o Resvd is reserved for future use.
- o 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.

3.1.4. 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 7:

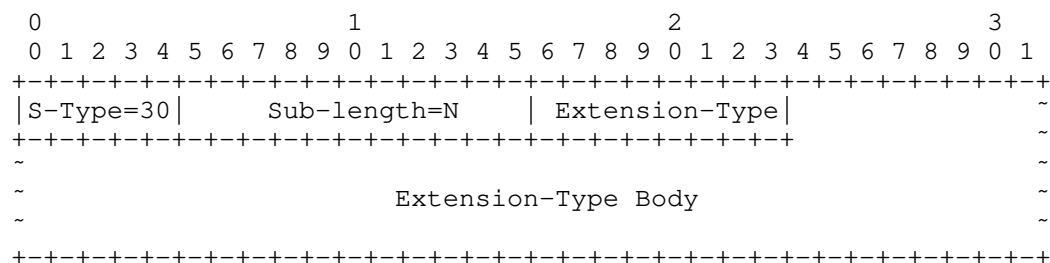


Figure 7: Sub-Type Extension

- o 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.
- o 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.
- o Extension-Type contains a 1 octet Sub-Type Extension value between 0 and 255.
- o Extension-Type Body contains an N-1 octet block with format defined by the given extension specification.

Extension-Type values 0 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.

4. IANA Considerations

The IANA is instructed to allocate a Type number TBD1 from the registry "IPv6 Neighbor Discovery Option Formats" for the OMNI option (see: Section 13 of [RFC4861]) as a provisional registration in accordance with Section 4.13 of [RFC8126].

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 for the OMNI option Sub-Type values registry are given below; future assignments are to be made through Expert Review [RFC8126].

Value	Sub-Type name	Reference
-----	-----	-----
0	Pad1	[RFCXXXX]
1	PadN	[RFCXXXX]
2	Interface Attributes (Type 1)	[RFCXXXX]
3-29	Unassigned	
30	Sub-Type Extension	[RFCXXXX]
31	Reserved	[RFCXXXX]

Figure 8: OMNI Option Sub-Type Values

5. Security Considerations

Security considerations for IPv6 [RFC8200] and IPv6 Neighbor Discovery [RFC4861] apply.

6. Acknowledgements

This document is aligned with the International Civil Aviation Organization (ICAO) Aeronautical Telecommunications Network (ATN) with Internet Protocol Services (ATN/IPS) development program.

This document is aligned with the IETF 6man (IPv6) working group.

7. References

7.1. Normative References

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

- [RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, DOI 10.17487/RFC3692, January 2004, <<https://www.rfc-editor.org/info/rfc3692>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

7.2. Informative References

- [ATN] Maiolla, V., "The OMNI Interface - An IPv6 Air/Ground Interface for Civil Aviation, IETF Liaison Statement #1676, <https://datatracker.ietf.org/liaison/1676/>", March 2020.
- [ATN-IPS] WG-I, ICAO., "ICAO Document 9896 (Manual on the Aeronautical Telecommunication Network (ATN) using Internet Protocol Suite (IPS) Standards and Protocol), Draft Edition 3 (work-in-progress)", December 2020.
- [I-D.templin-6man-omni-interface] Templin, F. and T. Whyman, "Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces", draft-templin-6man-omni-interface-69 (work in progress), January 2021.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.

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

RTGWG Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2022

F. Yang
M. Chen
T. Zhou
Huawei Technologies
July 12, 2021

Associated Channel over IPv6
draft-yang-rtgwg-ipv6-associated-channel-01

Abstract

This document introduces a control channel based on IPv6, called Associated CHannel over IPv6 (ACH6), that may carry different types of control and management messages. By using the associated channel, messages can be transmitted between network nodes to provide functions like path identification, OAM, automatic protection switchover signaling, resource reservation, etc., targeting to provide high quality SLA guarantees to IPv6 services.

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."

This Internet-Draft will expire on January 13, 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 publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Architecture of Associated Channel over IPv6	4
3.1. ACH6 Network Reference Model	4
3.2. ACH6 Processing	5
4. Format of Associated Channel over IPv6	5
4.1. Identification of ACH6	5
4.2. Carried Message of ACH6	6
4.3. ACH6 TLV Format	6
4.4. Encapsulation of ACH6 TLV in IPv6	7
4.4.1. Encapsulated in IPv6 Destination Options Header	7
4.4.2. Encapsulated in IPv6 Hop-by-Hop Options Header	7
4.4.3. Encapsulated in IPv6 Segment Routing Header	8
4.4.4. Encapsulated in Payload	9
4.5. Considerations	10
5. Applicability	10
5.1. Path Identification	10
5.2. OAM	10
5.3. Assist to Protection Switchover	11
6. IANA Considerations	11
7. Security Considerations	11
8. Acknowledgements	11
9. References	11
9.1. Normative References	11
9.2. Informative References	12
Authors' Addresses	12

1. Introduction

IPv6 is becoming widely accepted to provide connectivity in many new emerging scenarios, including Cloud Network convergence, Cloud-Cloud interconnection, 5G vertical industries, and Internet of Things etc. However, due to the best effort forwarding genes, native IP (for both IPv4 and IPv6) cannot provide the forwarding capabilities such as explicit path, control and management based on the forwarding path, to meet the requirements of services accordingly.

Generic Associated Channel (G-ACh) [RFC5586] introduces an associated control channel to MPLS to provide a set of maintenance functions, including OAM, performance monitoring, automatic protection switching and support of management to MPLS Sections, MPLS Label Switched Paths (LSPs), and MPLS pseudowires (PWs).

Triggered by MPLS G-ACh, to enhance the control and management capabilities to IPv6, this document introduces an associated channel to a specific IPv6 path, called Associated Channel over IPv6 (ACH6). Associated channel over IPv6 intends to create a control channel associated with the IPv6 data forwarding path for control and management purposes. By using the associated channel, messages can be transmitted between the network nodes to provide functions like path identification, OAM, automatic protection switchover signaling, resource reservation, etc., targeting to provide high quality SLA guarantees to services. To identify an IPv6 forwarding path, associated channel ID is also introduced.

This document defines an ACH6 architecture, a TLV-based format of ACH6, and discusses how ACH6 format can be encapsulated in IPv6 packet. It also discusses the applicability of ACH6 in IPv6 network.

2. Terminology

This document uses the terminology defined in [RFC8200] , and it also introduces the following new terms:

OAM: Operations, Administration, and Maintenance

SLA: Service Level Agreement

G-ACh: Generic Associated Channel

ACH6: Associated CHannel over IPv6

3. Architecture of Associated Channel over IPv6

3.1. ACH6 Network Reference Model

Figure 1 gives a network reference model of associated channel over IPv6. The key components in ACH6 network reference model include ACH6 Ingress Node, ACH6 Mid-Point Node, and ACH6 Egress Node. These nodes must be IPv6-capable node.

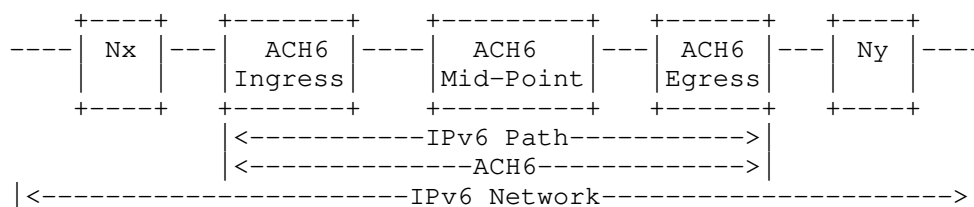


Figure 1 ACH6 Network Reference Model

In the network reference model,

Nx/Ny: IPv6 node, can be either a host or a router.

ACH6 Ingress Node: is the node indicates the entering of control and management channel over an IPv6 path, where control and management messages are generated and encapsulated.

ACH6 Mid-Point Node: is the node that has the capability to process control and management messages over an IPv6 path. For a strict explicit IPv6 path, all the IPv6 hop(s) forwarded from IPv6 source address to IPv6 destination address are mid-point node(s).

ACH6 Egress Node: is the node indicates the exiting of control and management channel over an IPv6 path, where control and management messages are recovered and delivered to control or management plane for further process.

IPv6 Path: a specific path from the source node to the destination node in IPv6 forwarding plane. An IPv6 path can be explicitly or implicitly represented by forwarding hops from IPv6 source node to IPv6 destination node.

ACH6: Associated Channel over IPv6

3.2. ACH6 Processing

Regarding to an IPv6 path, an ACH6 control channel is established to this specific IPv6 path for a required purpose. ACH6 ingress node acts as the IPv6 source node, and ACH6 egress node is the IPv6 destination node. ACH6 ingress node encapsulates control or management messages into IPv6 packets, identifies the specific channel type which carried messages belong to, and sends the IPv6 ACH6 packets to the destination of IPv6 path. The control and management messages can either piggyback with data packets, or generated and transmitted separately.

When ACH6 Mid-Point Node receives ACH6 IPv6 packets, it firstly recognizes ACH6 associated channel to interpret the control protocol, and then processes the messages. The processing of messages can include for example READ or/and WRITE, depending on the specification of protocols used in the associated channel. ACH6 Mid-Point Node needs to transmit the IPv6 packets carried with the original or modified ACH6 messages to the destination of IPv6 packet.

ACH6 Egress Node receives ACH6 IPv6 packets and recognizes itself as the destination. Based on the specific type of control protocol, the message is delivered to control or management planes for further process.

4. Format of Associated Channel over IPv6

An associated channel provides a control channel that carries at least one or more types of control and management messages. The type of message is not limited to any specific usage. The associated channel is specified by two pieces of information, including the identification of the associated channel and the carried message.

4.1. Identification of ACH6

The identification of associated channel, called Associated Channel ID, indicates the path where the packets of the associated channel are transmitted on. This identification also indicates the same path of the service forwarding path with which the associated channel is associated. When the Associated Channel ID is carried in the associated channel, ACH6 edge nodes and intermediated nodes should interpret it to identify the same IPv6 path.

The associated channel ID can be defined either globally unique or site local, or even link local. The Associated Channel ID can be self-generated, or designated from management plane, or advertised and allocated via control protocol.

4.2. Carried Message of ACH6

At least one control or management protocol messages are transmitted via associated channel over IPv6. When multiple protocols are running over an IPv6 path, messages of different protocols can be sent either in separate ACH6 TLVs in one ACH6 packet or in separate ACH6 packets with only one type of ACH6 TLV.

4.3. ACH6 TLV Format

An Associated CHannel over IPv6 (ACH6) TLV is designed to carry the identification and carried message of an associated channel. ACH6 TLV has the following format:

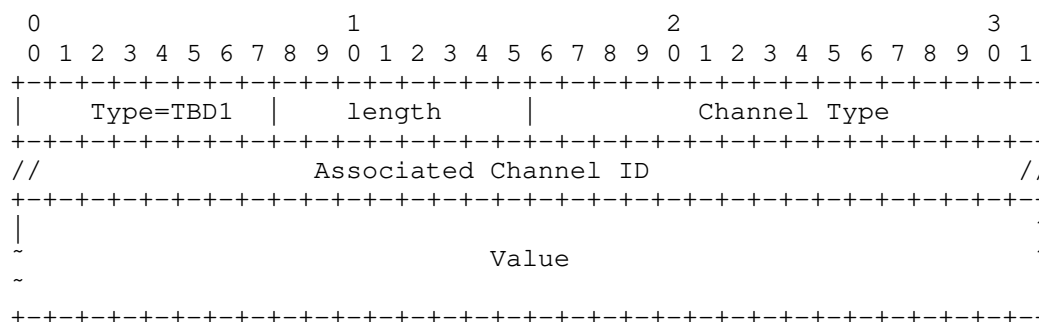


Figure 2 ACH6 TLV Format

Type: 8 bits, indicates it is an associated channel ACH6 TLV, and request a value assigned by IANA. The uniform type of TLV generalizes the applicability of ACH6 TLV to support various types of messages.

Length: 8 bits, defines the length of Value field in bytes.

Channel Type: is a 16-bit-length fixed portion of Value field. It indicates the specific type of messages carried in associated channel. Note that a new ACH6 TLV Channel Type Registry would be requested to IANA. In later documents which specify application protocols of associated channel, MUST also specify the applicable Channel Type field value assigned by IANA.

Associated Channel ID: indicates the identification of associated channel. The length is TBD.

Value: is a variable portion of Value field. It specifies the messages indicated by Channel Type and carried in associated channel.

Note that the Value field of ACH6 TLV MAY contain sub-TLVs to provide additional context information to ACH6 TLV.

4.4. Encapsulation of ACH6 TLV in IPv6

In the context of IPv6, ACH6 TLV can be encapsulated in different types of IPv6 extension headers, or as an independent IPv6 payload. Note that, no matter which way ACH6 TLV is applied, there is no semantic change to IPv6 extension headers.

4.4.1. Encapsulated in IPv6 Destination Options Header

ACH6 TLV can be encapsulated in IPv6 Destination Options Header as the TLV-encoded options. Figure 2 gives an example of an ACH6 TLV encapsulated in IPv6 Destination Options Header.

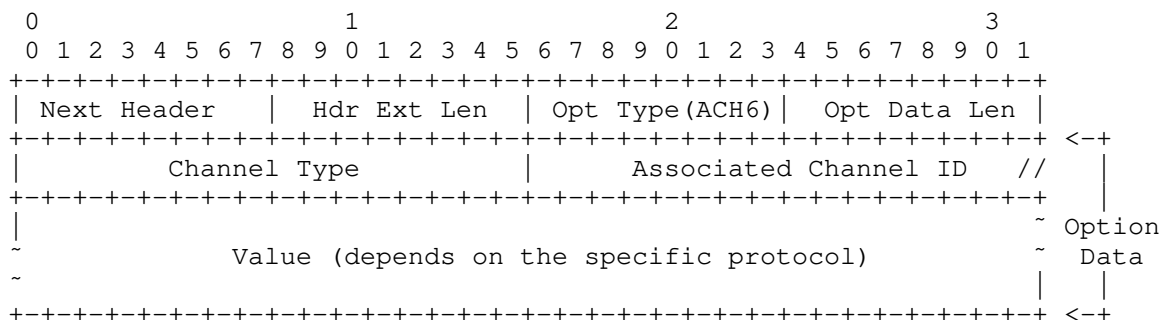


Figure 3 ACH6 TLV in IPv6 Destination Options Header

According to the note 1 and note 3 described in section 4.1 of[RFC8200], ACH6 TLV encapsulated in IPv6 Destination Options Header can provide two semantics of an associated channel. When only IPv6 Destination Options Header exists or IPv6 Destination Options Header exists after the Routing Header, an end to end associated channel is provided to transmit the messages between two endpoints. When both IPv6 Destination Options Header and Routing Header exist, and IPv6 Destination Options Header exists before the Routing Header, an associated channel is provided at network nodes of the first destination that appears in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header.

4.4.2. Encapsulated in IPv6 Hop-by-Hop Options Header

ACH6 TLV can be encapsulated in IPv6 Hop-by-Hop Options Header as the TLV-encoded options. Same option type numbering space is used for both Hop-by-Hop Options header and Destination Options header.

Similarly, the ACH6 TLV in IPv6 Hop-by-Hop Options Header shares the same encapsulation shown in Figure 3.

When it is encapsulated in IPv6 Hop-by-Hop Options Header, it provides an associated channel at every node along the forwarding path. ACH6 ingress node inserts the IPv6 HbH Option Header with ACH6 Option Type, every mid-point node examines, processes and transmits the IPv6 packet to next forwarding hop. ACH6 egress node receives the IPv6 packet as the destination node, ACH6 messages are processed and delivered to control or management plane for further usage. Processing is limited, can be READ and/or REWRITE.

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

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

4.4.3. Encapsulated in IPv6 Segment Routing Header

ACH6 TLV can be encapsulated in IPv6 Segment Routing Header, as SRH optional TLV. Figure 3 gives an example of an ACH6 TLV encapsulated in IPv6 Segment Routing Header.

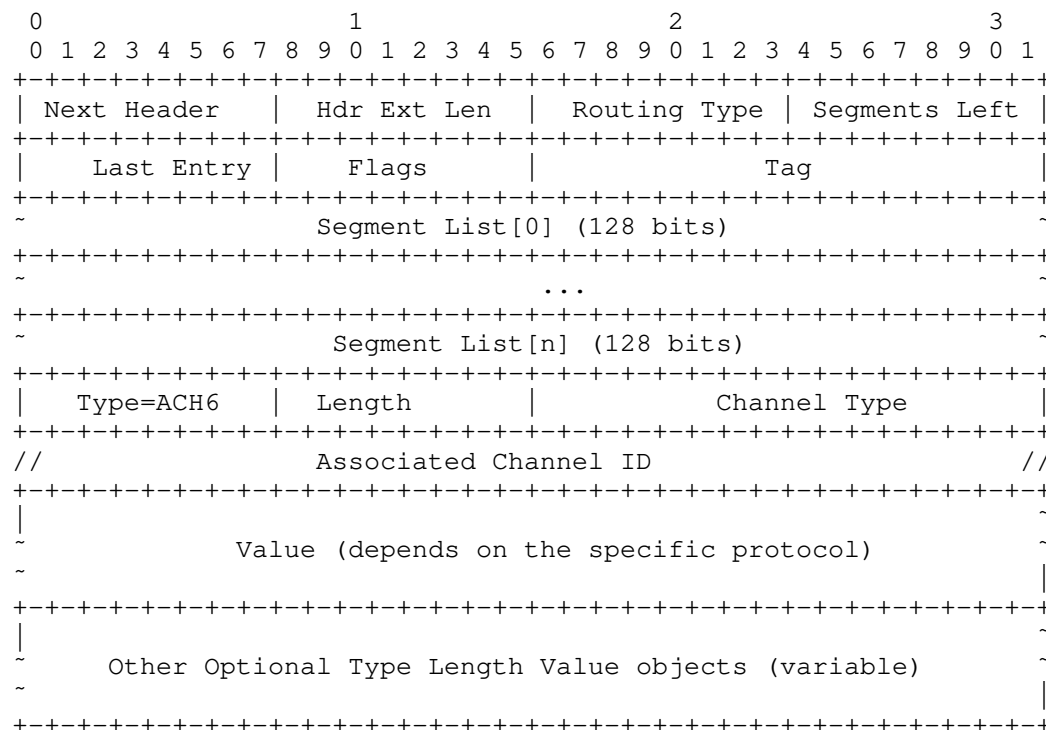


Figure 4 ACH6 TLV in IPv6 Segment Routing Header

When ACH6 TLV is encapsulated in IPv6 Segment Routing Header, it provides an associated channel at every SRv6 endpoints along the path.

4.4.4. Encapsulated in Payload

ACH6 TLV can also be encapsulated in the payload of an IPv6 packet. The term payload here means the octets after the IPv6 header and extension headers. A synthetic packet is created with the payload of messages. and transmitted in an associated channel. The synthetic packet can use the same routing information with service data whose associated channel it is associated. For example, synthetic packet can encapsulate the same segment list as the one used in IPv6 SRH of service data. If ACH6 TLV format is encapsulated in payload, TLV Type and Length can be omitted, a new codepoint of IP Protocol Numbers should be assigned.

4.5. Considerations

When ACH6 TLV is deployed in either IPv6 extension headers or payload in IPv6 networks, there are several considerations needs to be taken into account:

C1: MTU Increase

Given that ACH6 messages increases the packet size of IPv6 packet, it may face the risk of exceeding the PMTU. This problem can be solved by taking two things into considerations. Firstly, the mechanism of each protocol should clearly specify the maximum size limit of carried messages in one IPv6 packet. Secondly, operators or hosts who makes use of ACH6 to carry control and management messages should carefully design and ensure the addition of messages would not exceed the agreed PMTU.

C2: Processing of IPv6 Extension Header

Though IPv6 Extension Headers especially IPv6 Hop-by-Hop Option Header is not widely used in the Internet, there are some limited environments like Data Centers and Interconnections between Data Centers are experimentally using IPv6 Option Headers. It is worth to keep the possibility of carrying ACH6 messages as Option in IPv6 Extension Headers.

5. Applicability

5.1. Path Identification

In a native IPv6 network, packets are transmitted hop by hop, there is no way to identify an IPv6 forwarding path. The path needs to be identified when OAM or protection switchover is applied to the path.

5.2. OAM

OAM includes a group of functions such as connectivity verification, fault indication and detection, and performance measurement of loss and delay etc. For example, BFD defines a generic control packet format that can be encapsulated in different data planes to provide low-overhead and short-duration failure detection function. The format can also be encapsulated in ACH6 TLV as the option TLV of Destination Options Header, to provide the same connectivity verification and fault detect functions without introducing upper layer protocols. Another example is to encapsulate PDU formats of Ethernet OAM [ITU-T G.8013] in Value field of ACH6 TLV to provide a set of OAM functions. By using ACH6 TLV to carry OAM messages in an associated channel, different OAM functions can be easily integrated.

The OAM functions can be performed in either end-to-end or hop-by-hop mode. For example, signal degradation that happens on the intermediate node could be discovered and further indicated in associated channel to monitor the path status.

5.3. Assist to Protection Switchover

Linear protection [RFC6378] provides a very flexible protection mechanism in a mesh network because it can operate between any pair of endpoints. ACH6 TLV can be used to transmit the protection state control messages on an IPv6 forwarding path to provide the function of bidirectional protection switchover.

6. IANA Considerations

- o This document requests IANA to assign a codepoint of Destination Options and Hop-by-Hop Options.
- o This document requests IANA to assign a codepoint of Segment Routing Header TLVs to indicate ACH6 TLV.
- o This document request IANA to create a new registry of IPv6 ACH6 Channel Types to identify the usage of associated channel.

7. Security Considerations

TBD

8. Acknowledgements

TBD

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

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

9.2. Informative References

- [I-D.ietf-spring-srv6-path-segment] Li, C., Cheng, W., Chen, M., Dhody, D., and R. Gandhi, "Path Segment for SRv6 (Segment Routing in IPv6)", draft-ietf-spring-srv6-path-segment-00 (work in progress), November 2020.
- [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <<https://www.rfc-editor.org/info/rfc5586>>.
- [RFC6378] Weingarten, Y., Ed., Bryant, S., Osborne, E., Sprecher, N., and A. Fulignoli, Ed., "MPLS Transport Profile (MPLS-TP) Linear Protection", RFC 6378, DOI 10.17487/RFC6378, October 2011, <<https://www.rfc-editor.org/info/rfc6378>>.
- [RFC8402] Filts, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8754] Filts, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.

Authors' Addresses

Fan Yang
Huawei Technologies
Beijing
China

Email: shirley.yangfan@huawei.com

Mach(Guoyi) Chen
Huawei Technologies
Beijing
China

Email: mach.chen@huawei.com

Tianran Zhou
Huawei Technologies
Beijing
China

Email: zhoutianran@huawei.com