Workgroup: Network Working Group
Internet-Draft: draft-templin-6man-omni-67
Published: 28 June 2022
Intended Status: Informational
Expires: 30 December 2022
Authors: F. L. Templin, Ed.
The Boeing Company

Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces

Abstract

Mobile nodes (e.g., aircraft of various configurations, terrestrial vehicles, seagoing vessels, space systems, enterprise wireless devices, pedestrians with cell phones, etc.) communicate with networked correspondents over multiple access network data links and configure mobile routers to connect end user networks. A multilink virtual interface specification is presented that enables mobile nodes to coordinate with a network-based mobility service and/or with other mobile node peers. The virtual interface provides an adaptation layer service that also applies for more static deployments such as enterprise and home networks. 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 30 December 2022.

Table of Contents

1. Introduction

Mobile nodes (e.g., aircraft of various configurations, terrestrial vehicles, seagoing vessels, space systems, enterprise wireless devices, pedestrians with cellphones, etc.) configure mobile routers with multiple interface connections to wireless and/or wired-line data links. 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. The mobile router acts as a Client of a network-based Mobility Service (MS) by configuring a virtual interface over its underlay interface data link connections.

Each Client configures a virtual interface (termed the "Overlay Multilink Network Interface (OMNI)") as a thin layer over its underlay network interfaces (which may themselves connect to virtual or physical links). 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 underlay 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 adapted to diverse underlay interfaces with heterogeneous properties.

The OMNI interface connects to a virtual overlay known as the "OMNI link". The OMNI link multinet service spans one or more Internetworks that may include private-use infrastructures (e.g., enterprise networks) and/or the global public Internet itself. Together, OMNI and the OAL provide the foundational elements required to support the "6 M's of modern Internetworking", including:

1. Multilink - a Client's ability to coordinate multiple diverse underlay interfaces as a single logical unit (i.e., the OMNI interface) to achieve the required communications performance and reliability objectives.

2. Multinet - the ability to span the OMNI link over a segment routing topology with multiple diverse administrative domain network segments while maintaining seamless end-to-end communications between mobile Clients and correspondents such as air traffic controllers, fleet administrators, etc.

3. Mobility - a Client's ability to change network points of attachment (e.g., moving between wireless base stations) which may result in an underlay interface address change, but without disruptions to ongoing communication sessions with peers over the OMNI link.

4. Multicast - the ability to send a single network transmission that reaches multiple Clients belonging to the same interest group, but without disturbing other Clients not subscribed to the interest group.

5. Multihop - a mobile Client vehicle-to-vehicle relaying capability useful when multiple forwarding hops between vehicles may be necessary to "reach back" to an infrastructure access point connection to the OMNI link.

6. MTU assurance - the ability to deliver packets of various robust sizes between peers without loss due to a link size restriction, and to dynamically adjust packets sizes to achieve the optimal performance for each independent traffic flow.

Client OMNI interfaces interact with the MS and/or other OMNI nodes through IPv6 Neighbor Discovery (ND) control message exchanges [RFC4861]. The MS consists of a distributed set of service nodes (including Proxy/Servers and other infrastructure elements) that also configure OMNI interfaces. Automatic Extended Route Optimization (AERO) in particular provides a companion MS compatible with the OMNI architecture [I-D.templin-6man-aero]. AERO discusses details of ND message based route optimization, mobility management, and multinet traversal while the fundamental aspects of OMNI link operation are discussed in this document.

Each OMNI interface provides a multilink nexus for exchanging inbound and outbound traffic via selected underlay 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 assigned to the link and from which Mobile Network Prefixes (MNPs) are derived. If there are multiple OMNI links, the IP layer will see multiple OMNI interfaces.

Each Client receives an MNP through IPv6 ND control message exchanges with Proxy/Servers over Access Networks (ANETs) and/or open Internetworks (INETs). The Client sub-delegates the MNP to downstream-attached End-user Networks (ENETs) independently of the underlay interfaces selected for data transport. The Client acts as a fixed or mobile router on behalf of peers on its ENETs, and uses OMNI interface control messaging to coordinate with Hosts, Proxy/Servers and/or other Clients. The Client iterates its control

messaging over each of the OMNI interface's ANET/INET underlay interfaces in order to register each interface with the MS (see Section 15). The Client can also provide Proxy/Server-like services for a recursively nested chain of other Clients located in downstream-attached ENETs.

Clients 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 underlay interfaces and provides a nexus for Safety-Based Multilink (SBM) operation. The IP layer applies SBM routing to select a specific OMNI interface, then the selected OMNI interface applies Performance-Based Multilink (PBM) internally to select appropriate underlay interfaces. Applications select SBM topologies based on IP layer Segment Routing [RFC8402], while each OMNI interface orchestrates PBM internally based on OMNI layer Segment Routing.

OMNI provides a link model suitable for a wide range of use cases. For example, 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 control messages over OMNI interfaces. The operation of both IP protocol versions (i.e., IPv4 [RFC0791] and IPv6 [RFC8200]) is specified as the network layer data plane, while OMNI interfaces use IPv6 ND messaging in the control plane independently of the data plane protocol(s). OMNI interfaces also provide an OAL based on encapsulation and fragmentation over heterogeneous underlay interfaces as an adaptation sublayer between L3 and L2. Both OMNI and the OAL are specified in detail throughout the remainder of this document.

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. Hosts, Clients and Proxy/Servers that implement IPv6 ND maintain per-neighbor state in Neighbor Cache Entries (NCEs). Each NCE is indexed by the neighbor's network layer address(es) while the neighbor's OAL encapsulation address provides context for Identification verification.

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 terms Host, Client and Proxy/Server are intentionally capitalized to denote a node of that particular node type that also configures an OMNI interface and engages the OMNI Adaptation Layer.

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

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

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

Adaptation layer
: A mid-layer that adapts L3 to a diverse collection of L2 underlay interfaces and their encapsulations. (No layer number is assigned, since numbering was an artifact of the legacy reference model that need not carry forward in the modern architecture.) The adaptation layer sees the upper layer as "L3" and sees all lower layer encapsulations as "L2 encapsulations", which may include UDP, IP and true link-layer (e.g., Ethernet, etc.) headers.

Access Network (ANET)
: a connected network region (e.g., an aviation radio access network, satellite service provider network, cellular operator network, WiFi network, etc.) that connects Clients to the Mobility Service. 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 Client's point of connection and the nearest Proxy/Server.

Internetwork (INET)
a connected network region with a coherent IP addressing plan that provides transit forwarding services between ANETs and/or OMNI nodes that coordinate with the Mobility Service over unprotected media. Since physical and/or data link level security cannot always be assumed, security must be applied by upper layers if necessary. The global public Internet itself is an example.

End-user Network (ENET)
a simple or complex "downstream" network that travels with the Client as a single logical unit. The ENET could be as simple as a single link connecting a single Host, or as complex as a large network with many links, routers, bridges and Hosts. The ENET could also provide an "upstream" link in a recursively-descending chain of additional Clients and ENETs. In this way, an ENET of an upstream Client is seen as the ANET of a downstream Client.

{A,I,E}NET interface
a Client's attachment to a link in an {A,I,E}NET.

***NET**
a "wildcard" term used when a given specification applies equally to both ANET/INET cases. From the Client's perspective, *NET interfaces are "upstream" interfaces that connect the Client to the Mobility Service, while ENET interfaces are "downstream" interfaces that the Client uses to connect downstream ENETs, Hosts and/or other Clients.

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

OMNI link
a Non-Broadcast, Multiple Access (NBMA) virtual overlay configured over one or more INETs and their connected ANETs/ENETs. An OMNI link may comprise multiple distinct "segments" joined by L2 forwarding devices the same as for any link; the addressing plans in each segment may be mutually exclusive and managed by different administrative entities. Proxy/Servers and other infrastructure elements extend the link to support communications between Clients as single-hop neighbors.

OMNI interface

a node's attachment to an OMNI link, and configured over one or more underlay interfaces. If there are multiple OMNI links in an OMNI domain, a separate OMNI interface is configured for each link. The OMNI interface configures a Maximum Transmission Unit (MTU) and a Maximum Reassembly Unit (MRU) the same as any interface.

OMNI Adaptation Layer (OAL)

an OMNI interface sublayer service that encapsulates original IP packets admitted into the interface in an IPv6 header and/or subjects them to fragmentation and reassembly. The OAL is also responsible for generating MTU-related control messages as necessary, and for providing addressing context for OMNI link SRT traversal. The OAL presents a new layer in the Internet architecture known simply as the "adaptation layer".

Host

an end user device that extends the OMNI link over an ENET interface serviced by a Client. (As an implementation matter, the Host either assigns the same IP address from the ENET (underlay) interface to an (overlay) OMNI interface, or configures an OMNI-like function as a virtual sublayer of the ENET interface itself.) The IP addresses assigned to each Host ENET interface remain stable even if the Client's upstream *NET interface connections change.

Client

a network platform/device mobile router that configures one or more OMNI interfaces over distinct sets of underlay interfaces grouped as logical OMNI link units. The Client coordinates with the Mobility Service via upstream networks over *NET interfaces, and provides Proxy/Server services for Hosts and other Clients on ENET interface downstream networks. The Client's *NET interface addresses and performance characteristics may change over time (e.g., due to node mobility, link quality, etc.) while downstream-attached Hosts and other Clients see the ENET as a stable ANET.

Proxy/Server

a segment routing topology edge node that configures an OMNI interface and connects Clients to the Mobility Service. As a server, the Proxy/Server responds directly to some Client IPv6 ND messages. As a proxy, the Proxy/Server forwards other Client IPv6 ND messages to other Proxy/Servers and Clients. As a router, the Proxy/Server provides a forwarding service for ordinary data packets that may be essential in some environments and a last resort in others. Proxy/Servers at ANET boundaries configure both

an ANET downstream interface and *NET upstream interface, while INET-based Proxy/Servers configure only an INET interface.

First-Hop Segment (FHS) Proxy/Server
: a Proxy/Server connected to the source Client's *NET that forwards packets sent by the source into the segment routing topology. FHS Proxy/Servers also act as intermediate forwarding nodes to facilitate RS/RA exchanges between Clients and Hub Proxy/Servers.

Last-Hop Segment (LHS) Proxy/Server
: a Proxy/Server connected to the target Client's *NET that forwards packets received from the segment routing topology to the target.

Hub Proxy/Server
: a single Proxy/Server selected by the Client that provides a designated router service for all of the Client's*NET underlay networks. Since all Proxy/Servers provide equivalent services, Clients normally select the first FHS Proxy/Server they coordinate with to serve as the Hub. However, the Hub can also be any available Proxy/Server for the OMNI link, i.e., and not necessarily one of the Client's FHS Proxy/Servers.

Segment Routing Topology (SRT)
: a multinet forwarding region configured over one or more INETs between the FHS Proxy/Server and LHS Proxy/Server. The SRT spans the OMNI link on behalf of source/target Client pairs using segment routing in a manner outside the scope of this document (see: [I-D.templin-6man-aero]).

Mobility Service (MS)
: a mobile routing service that tracks Client movements and ensures that Clients remain continuously reachable even across mobility events. The MS consists of the set of all Proxy/Servers and any other OMNI link supporting infrastructure nodes. Specific MS details are out of scope for this document, with an example found in [I-D.templin-6man-aero].

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 also 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 Client. Clients receive MNPs from Proxy/Servers and sub-delegate them to routers, Hosts and other Clients located in ENETs.

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 an IPv6 header (i.e., the OAL header) then submitted for OAL fragmentation and reassembly.

OAL fragment
: a portion of an OAL packet following fragmentation but prior to encapsulation, or following 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 L2 encapsulation or prior to L2 decapsulation. OAL sources and destinations exchange carrier packets over underlay 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 L2 headers of the first hop network and replacing them with new L2 headers for the next hop network. (The term "carrier" honors agents of the service postulated by [RFC1149] and [RFC6214].)

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 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 L2 encapsulation headers of carrier packets received

from a first segment, then re-encapsulates the carrier packets in new L2 headers and forwards them into the next segment. OAL intermediate nodes decrement the OAL Hop Limit during forwarding, and discard the packet if the Hop Limit reaches 0. OAL intermediate nodes do not decrement the TTL/Hop Limit of the original IP packet.

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

Interface Identifier (IID)
the least significant 64 bits of an IPv6 address, as specified in the IPv6 addressing architecture [RFC4291].

Link Local Address (LLA)
an IPv6 address beginning with fe80::/64 per the IPv6 addressing architecture [RFC4291] and with either a 64-bit MNP (LLA-MNP) or a 56-bit random value (LLA-RND) encoded in the IID as specified in Section 8.

Unique Local Address (ULA)
an IPv6 address beginning with fd00::/8 followed by a 40-bit Global ID followed by a 16-bit Subnet ID per [RFC4193] and with either a 64-bit MNP (ULA-MNP) or a 56-bit random value (ULA-RND) encoded in the IID as specified in Section 9. (Note that [RFC4193] specifies a second form of ULAs based on the prefix fc00::/8, which are referred to as "ULA-C" throughout this document to distinguish them from the ULAs defined here.)

Temporary Local Address (TLA)
a ULA beginning with fd00::/16 followed by a 48-bit randomly-initialized value followed by an MNP-based (TLA-MNP) or random (TLA-RND) IID as specified in Section 9. Clients use TLAs to bootstrap autoconfiguration in the presence of OMNI link infrastructure or for sustained communications in the absence of infrastructure. (Note that in some environments Clients can instead use a (Hierarchical) Host Identity Tag ((H)HIT) instead of a TLA - see: Section 22.)

eXtended Local Address (XLA)
a TLA beginning with fd00::/64 followed by an MNP-based (XLA-MNP) or random (XLA-RND) IID as specified in Section 9. An XLA is simply a TLA with an all-0 48-bit value following fd00::/16, and can be used to supply a "wildcard match" for IPv6 ND cache entries, a routing table entry for the OMNI link routing system, etc. (Note that XLAs can also be statelessly formed from LLAs (and vice-versa) simply by inverting prefix bits 7 and 8.)

Multilink

a Client OMNI interface's manner of managing multiple diverse *NET underlay interfaces as a single logical unit. The OMNI interface provides a single unified interface to upper layers, while underlay interface selections are performed on a per-packet basis considering traffic selectors such as DSCP, flow label, application policy, signal quality, cost, etc. Multilink selections are coordinated in both the outbound and inbound directions based on source/target underlay interface pairs.

Multinet

an intermediate node's manner of spanning multiple diverse IP Internetwork and/or private enterprise network "segments" at the OAL layer below IP. Through intermediate node concatenation of SRT network segments, multiple diverse Internetworks (such as the global public IPv4 and IPv6 Internets) can serve as transit segments in an end-to-end L2 forwarding path. This OAL concatenation capability provides benefits such as supporting IPv4/IPv6 transition and coexistence, joining multiple diverse operator networks into a cooperative single service network, etc. See: [I-D.templin-6man-aero] for further information.

Multihop

an iterative relaying of IP packets between Client's over an OMNI underlay interface technology (such as omnidirectional wireless) without support of fixed infrastructure. Multihop services entail Client-to-Client relaying within a Mobile/Vehicular Ad-hoc Network (MANET/VANET) for Vehicle-to-Vehicle (V2V) communications and/or for Vehicle-to-Infrastructure (V2I) "range extension" where Clients within range of communications infrastructure elements provide forwarding services for other Clients.

Mobility

any action that results in a change to a Client underlay interface address. The change could be due to, e.g., a handover to a new wireless base station, loss of link due to signal fading, an actual physical node movement, etc.

Safety-Based Multilink (SBM)

A means for ensuring fault tolerance through redundancy by connecting multiple OMNI interfaces within the same domain to

independent routing topologies (i.e., multiple independent OMNI links).

Performance Based Multilink (PBM)
A means for selecting one or more underlay 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. All OMNI links within the same domain configure, advertise and respond to the same OMNI IPv6 Anycast address(es).

AERO Forwarding Information Base (AFIB)
A multilink forwarding table on each OAL source, destination and intermediate node that includes AERO Forwarding Vectors (AFV) with both next hop forwarding instructions and context for reconstructing compressed headers for specific underlay interface pairs used to communicate with peers. See: [I-D.templin-6man-aero] for further discussion.

AERO Forwarding Vector (AFV)
An AFIB entry that includes soft state for each underlay interface pairwise communication session between peers. AFVs are identified by both a next-hop and previous-hop AFV Index (AFVI), with the next-hop established based on an IPv6 ND solicitation and the previous hop established based on the solicited IPv6 ND advertisement response. See: [I-D.templin-6man-aero] for further discussion.

AERO Forwarding Vector Index (AVFI)
A locally-unique 4 octet value that an OAL node generates when it creates an AFV, then advertises to either next-hop or previous-hop nodes. OAL intermediate nodes assign two distinct AFVIs for each AFV and advertise one to next-hops and the other to previous-hops. OAL end systems assign and advertise a single AFVI. See: [I-D.templin-6man-aero] for further discussion.

IP Jumbogram
an IPv4 or IPv6 packet with a Jumbo Payload option that includes a 32-bit length field to be used instead of the 16-bit {Total, Payload} Length field (see: Section 5.1). For IPv4, the Total Length field must be set to the length of the IPv4 header only. For IPv6, the Payload Length must be set to 0.

IP Parcel
a special form of an IP Jumbogram with a segment length value included in the {Total, Payload} Length field and also with a Jumbo Payload option (see: Section 5.2).

L2 encapsulation
the OAL encapsulation of a packet in an outer header or headers that can be routed within the scope of the local {A,I,E}NET underlay network partition. Common L2 encapsulation combinations include UDP/IP/Ethernet, etc.

L2 address (L2ADDR)
an address that appears in the OAL L2 encapsulation for an underlay interface and also in IPv6 ND message OMNI options. L2ADDR can be either an IP address for IP encapsulations or an IEEE EUI address [EUI] for direct data link encapsulation. (When UDP/IP encapsulation is used, the UDP port number is considered an ancillary extension of the IP L2ADDR.)

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 underlay 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 underlay interfaces that appear as L2 communication channels in the architecture.

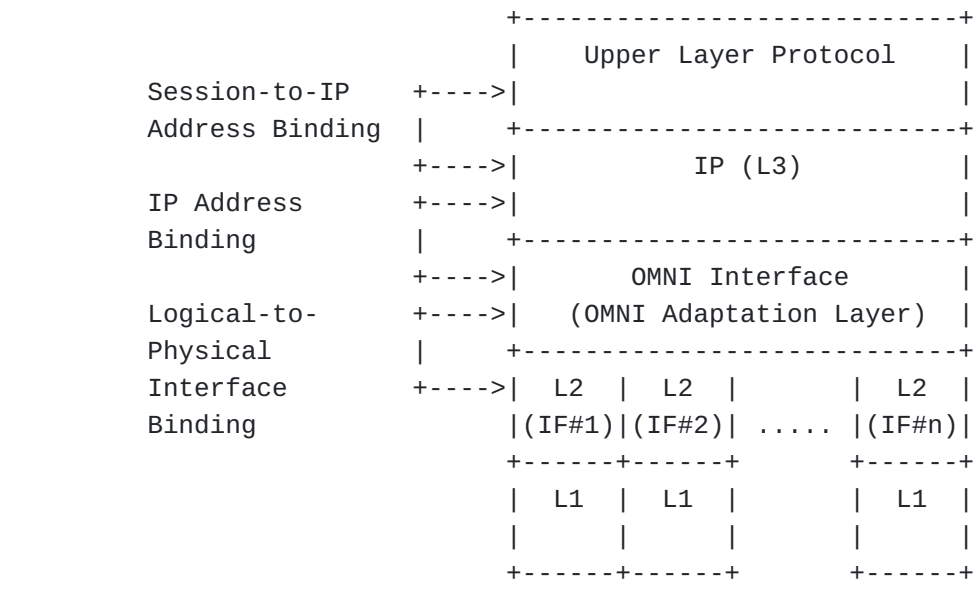


Figure 1: OMNI Interface Architectural Layering Model

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

*ANET interfaces connect to a protected and secured ANET that is separated from the open INET by Proxy/Servers. The ANET interface may be either on the same L2 link segment as a Proxy/Server, or separated from a Proxy/Server by multiple IP hops. (Note that NATs may appear internally within an ANET or on the Proxy/Server itself and may require NAT traversal the same as for the INET case.)

*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 configure private IP addresses and connect to a private network behind one or more NATs with the outermost NAT providing INET access.

*ENET interfaces connect a Client's downstream-attached networks, where the Client provides forwarding services for ENET Host and Client communications to remote peers. An ENET may be as simple as a small stub network that travels with a mobile Client (e.g., an Internet-of-Things) to as complex as a large private enterprise network that the Client connects to a larger ANET or INET. Downstream-attached Hosts and Clients see the ENET as an ANET and see the (upstream) Client as a Proxy/Server.

*VPNed interfaces use security encapsulation over an underlay network to a Client or Proxy/Server acting as a Virtual Private Network (VPN) gateway. Other than the link-layer encapsulation

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

*Direct (aka "point-to-point") interfaces connect directly to a Client or Proxy/Server without crossing any networked 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 underlay network headers (e.g., UDP/IP, IP-only, Ethernet-only, etc.) to create L2-encapsulated "carrier packets" for transmission over underlay interfaces. The target OMNI interface receives the carrier packets from underlay interfaces and discards the L2 encapsulation 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. If the OAL fragments are addressed to another node, the OMNI interface instead acts as an "OAL intermediate node" by re-encapsulating the carrier packets in new underlay network L2 headers and forwarding them over an underlay 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 provide a virtual bridging service that joins the segments of a (multinet) Segment Routing Topology (SRT).

The OMNI interface can forward original IP packets over underlay interfaces while including/omitting various lower layer encapsulations including OAL, UDP, IP and Ethernet (ETH) or other link-layer header. The network layer can also access the underlay interfaces directly while bypassing the OMNI interface entirely when necessary. This architectural flexibility may be beneficial for underlay 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 underlay interfaces without invoking the OAL can only reach peers located on the same OMNI link segment. Source Clients can instead use the OAL to coordinate with target Clients in the same or different OMNI link segments by sending initial carrier packets to a First-Hop Segment (FHS) Proxy/Server. The FHS Proxy/Sever then forwards the packets into the SRT spanning tree, which transports them to a Last-Hop Segment (LHS) Proxy/Server for the target Client.

Original IP packets sent directly over underlay 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 underlay interface that exceed the underlay 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. Imaginary vertical lines drawn between the Network Layer and Underlay interfaces in the figure denote the encapsulation/decapsulation layering combinations possible. Common combinations include IP-only (i.e., direct access to underlay interfaces with or without using the OMNI interface), IP/IP, IP/UDP/IP, IP/UDP/IP/ETH(ERNET), IP/OAL/UDP/IP, IP/OAL/UDP/ETH, etc.

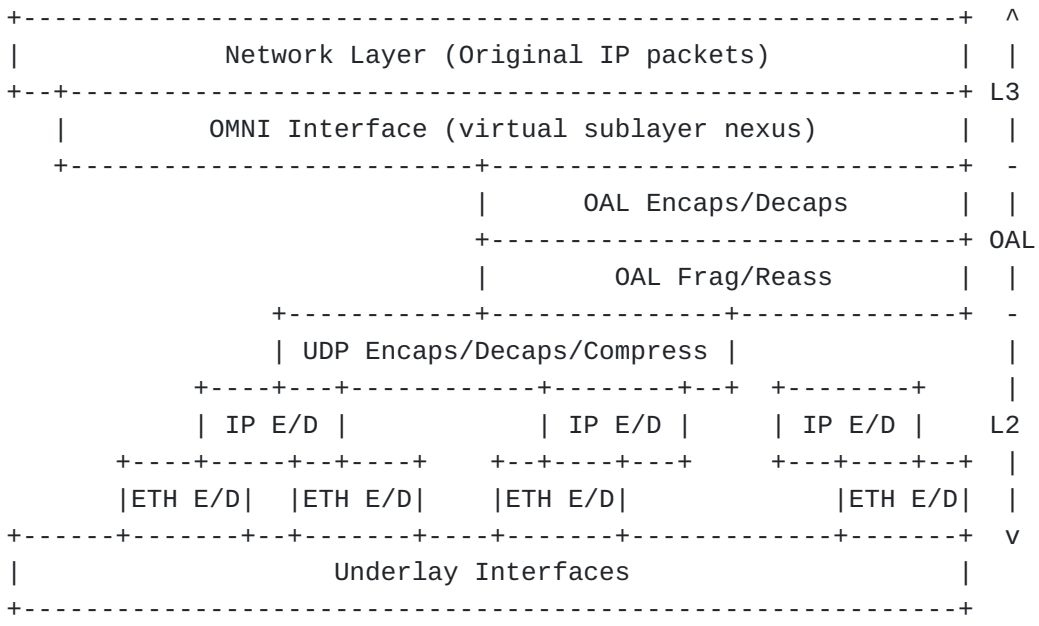


Figure 2: OMNI Interface Layering

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

*Clients receive MNPs from the MS, and coordinate with the MS through IPv6 ND message exchanges with Proxy/Servers. Clients use the MNP to construct a unique Link-Local Address (LLA-MNP) through the algorithmic derivation specified in Section 8 and assign the LLA to the OMNI interface. Since LLA-MNPs are uniquely derived from an MNP, no Duplicate Address Detection (DAD) or Multicast Listener Discovery (MLD) messaging is necessary.

*since Temporary ULAs with random IIDs (TLA-RNDs) are statistically unique, they can be used without DAD until an MNP is obtained.

*underlay interfaces on the same L2 link segment as a Proxy/Server do not require any L3 addresses (i.e., not even link-local) in environments where communications are coordinated entirely over the OMNI interface.

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

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

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

*the OMNI interface allows multinet traversal over the SRT when communications across different administrative domain network segments are necessary. This mode of operation would not be possible via direct communications over the underlay interfaces themselves.

*the OAL supports lossless and adaptive path MTU mitigations not available for communications directly over the underlay interfaces themselves. The OAL supports "packing" of multiple IP payload packets within a single OAL "super-packet" and also supports transmission of IP packets and parcels of all sizes up to and including Jumbograms.

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

*L3 sees the OMNI interface as a point of connection to the OMNI link; if there are multiple OMNI links, L3 will see multiple OMNI interfaces.

*Multiple independent OMNI interfaces can be used for increased fault tolerance through Safety-Based Multilink (SBM), with Performance-Based Multilink (PBM) applied within each interface.

* Multiple independent OMNI links can be joined together into a single link without requiring renumbering of infrastructure elements, since the ULAs assigned to the different links will be mutually exclusive.

* the OMNI/OAL model supports transmission of a new form of IP packets known as "IP Parcels" that improve performance and efficiency for both upper layer protocols and networked paths.

Note that even when the OMNI virtual interface is present, applications can still access underlay 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 OMNI interface is configured over an underlay IP interface, applications can still invoke intra-network IP communications directly over the underlay interface as long as the communicating endpoints are not subject to mobility dynamics.

Figure 3 depicts the architectural model for a source Client with an attached ENET connecting to the OMNI link via multiple independent ANETs/INETs (i.e., *NETs). The Client's OMNI interface sends IPv6 ND solicitation messages over available *NET underlay interfaces using any necessary L2 encapsulations. The IPv6 ND messages traverse the *NETs until they reach an FHS Proxy/Server (FHS#1, FHS#2, ..., FHS#n), which returns an IPv6 ND advertisement message and/or forwards a proxyed version of the message over the SRT to an LHS Proxy/Server near the target Client (LHS#1, LHS#2, ..., LHS#m). The Hop Limit in IPv6 ND messages is not decremented due to encapsulation; hence, the source and target Client OMNI interfaces appear to be attached to a common link.

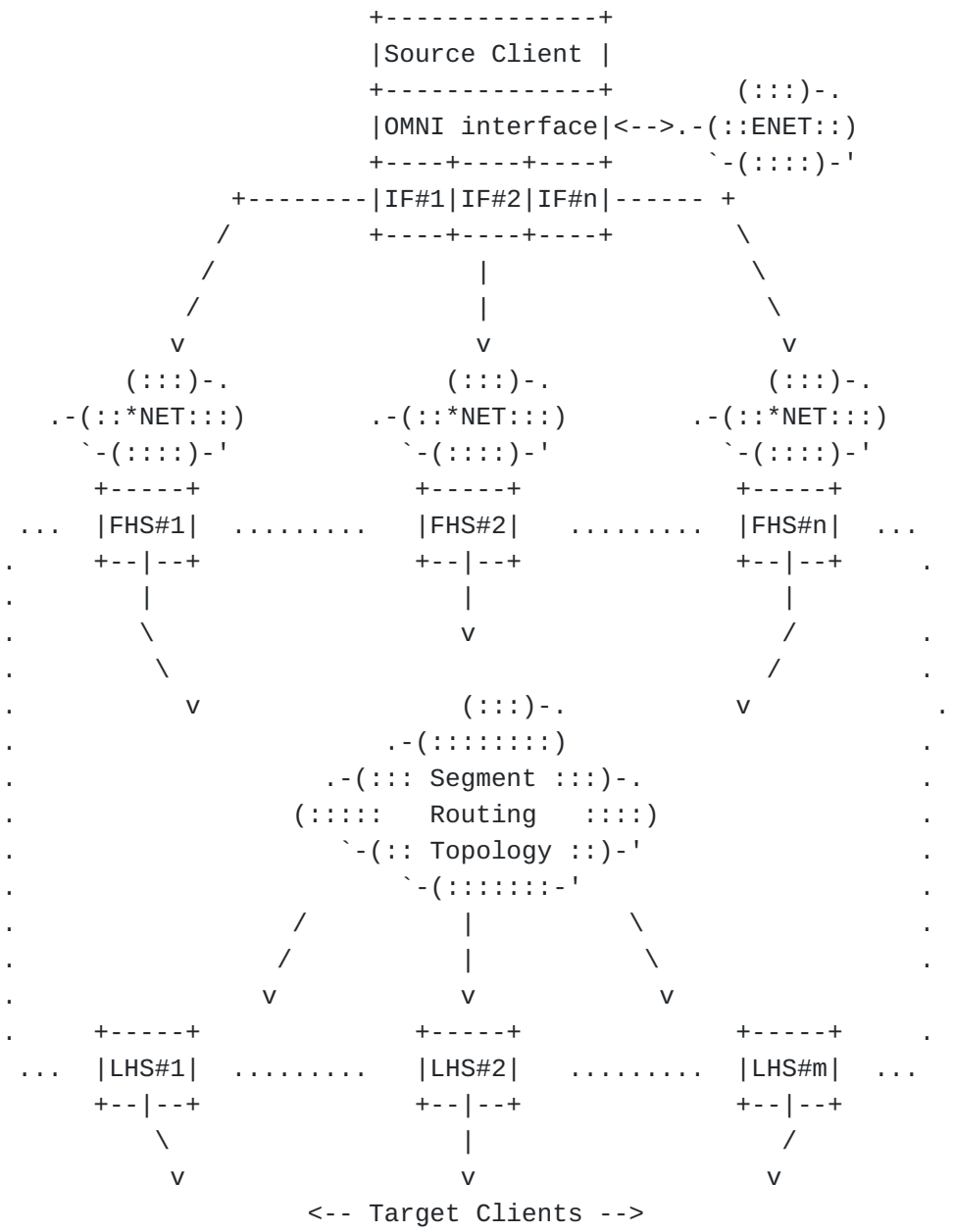


Figure 3: Source/Target Client Coordination over the OMNI Link

After the initial IPv6 ND message exchange, the source Client (as well as any nodes on its attached ENETs) can send packets to the target Client over the OMNI interface. OMNI interface multilink services will forward the packets via FHS Proxy/Servers for the correct underlay *NETs. The FHS Proxy/Server then forwards the packets over the SRT which delivers them to an LHS Proxy/Server, and the LHS Proxy/Server in turn forwards them to the target Client. (Note that when the source and target Client are on the same SRT segment, the FHS and LHS Proxy/Servers may be one and the same.)

Clients select a Hub Proxy/Server (not shown in the figure), which will often be one of their FHS Proxy/Servers but could also be any

Proxy/Server on the OMNI link. Clients then register all of their *NET underlay interfaces with the Hub Proxy/Server via the FHS Proxy/Server in a pure proxy role. The Hub Proxy/Server then provides a designated router service for the Client, and the Client can quickly migrate to a new Hub Proxy/Server if the first becomes unresponsive.

Clients therefore use Proxy/Servers as gateways into the SRT to reach OMNI link correspondents via a spanning tree established in a manner outside the scope of this document. Proxy/Servers forward critical MS control messages via the secured spanning tree and forward other messages via the unsecured spanning tree (see Security Considerations). When route optimization is applied as discussed in [I-D.templin-6man-aero], Clients can instead forward directly to SRT intermediate nodes (or directly to correspondents in the same SRT segment) to reduce Proxy/Server load.

Note: while not shown in the figure, a Client's ENET may connect many additional Hosts and even other Clients in a recursive extension of the OMNI link. This OMNI virtual link extension will be discussed more fully throughout the document.

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

IPv6 underlay interfaces are REQUIRED to configure a minimum MTU of 1280 octets and a minimum MRU of 1500 octets [RFC8200]. Therefore, the minimum IPv6 path MTU is 1280 octets 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 octets 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 octets and larger still if it knows the destination configures a larger MRU, this does not affect the minimum IPv6 path MTU.)

IPv4 underlay interfaces are REQUIRED to configure a minimum MTU of 68 octets [RFC0791] and a minimum MRU of 576 octets [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 octets 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 octets, and SHOULD set DF to 1 in larger carrier packets unless it has a way to determine the encapsulation destination MRU and has carefully considered the issues discussed in Section 6.12.

When the network layer admits an original IP packet into the OMNI interface the OAL prepends an IPv6 encapsulation header (see: Section 6) where the 16-bit Payload Length field limits the maximum-sized original IP packet to (2**16 -1) = 65535 octets; this is also the maximum size that the OAL can accommodate with IPv6 fragmentation. The OMNI interface therefore sets an MTU and MRU of 65535 octets to support assured delivery of original packets no larger than this size even if IPv6 fragmentation is required. (The OMNI interface MAY set a larger MTU to support best-effort delivery for larger packets; see below.) The OMNI interface then employs the OAL as an encapsulation sublayer service to transform original IP packets into OAL packets/fragments, and the OAL in turn uses underlay network encapsulation to forward carrier packets over underlay interfaces (see: Section 6).

5.1. Jumbograms

While the maximum-sized original IP packet that the OAL can accommodate using IPv6 fragmentation is 65535 octets, OMNI interfaces can forward still larger IPv6 packets as OAL "atomic fragments" through the application of IPv6 Jumbograms [RFC2675]. For such larger packets, the OMNI interface performs OAL encapsulation by appending an IPv6 header followed by an 8-octet Hop-By-Hop header with Jumbo Payload option followed by a Routing Header of no more than 40-octets (if necessary) and finally followed by an 8-octet Fragment Header.

Since the Jumbo Payload option includes a 32-bit length field, OMNI interfaces can therefore configure a larger IP MTU up to a maximum of ((2**32 - 1) - 8 - 40 - 8) = 4294967239 octets. In that case, the OAL will still provide original IP packets no larger than 65535 with an IPv6 fragmentation-based assured delivery service while larger IP packets will receive a best-effort delivery service as atomic fragments (note that the OAL destination is permitted to accept atomic fragments that exceed the OMNI interface MRU).

The OAL source forwards jumbo atomic fragments under the assumption that upper and lower layers will employ sufficient integrity assurance, noting that commonly-used 32-bit CRCs may be inadequate for these larger sizes [CRC]. If the packet is dropped along the

path to the OAL destination, the OAL source must arrange to return a PTB "hard error" to the original source Section 6.8.

This document notes that a Jumbogram service for IPv4 is also specified in [I-D.templin-intarea-parcels], where all OMNI link aspects of the service are conducted in a similar fashion as for IPv6 above.

5.2. IPv6 Parcels

As specified in [I-D.templin-intarea-parcels], an IP Parcel is a variation of the IP Jumbogram construction beginning with an IP header with the length of the first upper layer protocol segment in the {Total, Payload} Length field, but with a Jumbo Payload option with a length that may be the same as or larger than the length in the IP header. The differences in these lengths determines the size and number of upper layer protocol segments within the parcel.

The IP Parcel format and transmission/reception procedures for OMNI interfaces are specified in Section 6.14. End systems that implement either the full OMNI interface (i.e., Clients) or enough of the OAL to process parcels (i.e., Hosts) are permitted to exchange parcels with consenting peers.

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 underlay interfaces, the OMNI Adaptation Layer (OAL) acting as the OAL source applies encapsulation to form OAL packets subject to fragmentation producing OAL fragments suitable for L2 encapsulation and transmission as carrier packets over underlay interfaces as described in Section 6.1.

These carrier packets travel over one or more underlay networks spanned by OAL intermediate nodes in the SRT, which re-encapsulate by removing the L2 headers of the first underlay network and appending L2 headers appropriate for the next underlay network in succession. (This process supports the multinet concatenation capability needed for joining multiple diverse networks.) 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 L2 headers and reassembles the resulting OAL fragments (if necessary) into an OAL packet as described in Section 6.3. The OAL destination next decapsulates the OAL packet to obtain the original IP packet then delivers the original IP packet to the network layer. The OAL source may be either the source Client or its FHS Proxy/Server, while the OAL destination may be either the LHS Proxy/Server

or the target Client. Proxy/Servers (and SRT Gateways as discussed in [I-D.templin-6man-aero]) may also serve as OAL intermediate nodes.

The OAL presents an OMNI sublayer abstraction similar to ATM Adaptation Layer 5 (AAL5). Unlike AAL5 which performs segmentation and reassembly with fixed-length 53 octet cells over ATM networks, however, the OAL uses IPv6 encapsulation, fragmentation and reassembly with larger variable-length cells over heterogeneous underlay networks. Detailed operations of the OAL are specified 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 creates an "OAL packet" by prepending an IPv6 OAL encapsulation header per [RFC2473] but does not decrement the Hop Limit/TTL of the original IP packet since encapsulation occurs at a layer below IP forwarding. The OAL source copies the "Type of Service/Traffic Class" [RFC2983] and "Explicit Congestion Notification (ECN)" [RFC3168] values in the original packet's IP header into the corresponding fields in the OAL header, then sets the OAL header "Flow Label" as specified in [RFC6438]. The OAL source finally sets the OAL header IPv6 Payload Length to the length of the original IP packet and sets Hop Limit to a value that MUST NOT be larger than 63 yet is still sufficiently large to enable loop-free forwarding over multiple concatenated OMNI link intermediate hops.

The OAL next selects OAL packet source and destination addresses. Client OMNI interfaces set the OAL source address to a Unique Local Address (ULA) based on the Mobile Network Prefix (ULA-MNP). When a Client OMNI interface does not (yet) have a ULA prefix and/or an MNP suffix, it can instead use a Temporary ULA (TLA) (or a (Hierarchical) Host Identity Tag ((H)HIT - see: Section 22) as an OAL address. Finally, when the Client needs to express its MNP outside the context of a specific ULA prefix, it can use an eXtended ULA (XLA). Proxy/Server OMNI interfaces instead set the source address to a Random ULA (ULA-RND) (see: Section 9), but also process packets with anycast and/or multicast OAL addresses that they are configured to recognize.)

The OAL source next selects a 32-bit OAL packet Identification value as specified in Section 6.6. The OAL then calculates a 2-octet OAL checksum using the algorithm specified in Appendix A. The OAL source calculates the checksum over the OAL packet beginning with a pseudo-header of the OAL header similar to that found in Section 8.1 of [RFC8200], then extending over the entire length of the original IP packet. The OAL pseudo-header is formed as shown in Figure 4:

Figure 4: OAL Pseudo-Header

After calculating the checksum, the OAL source next fragments the OAL packet if necessary while assuming the IPv4 minimum path MTU (i.e., 576 octets) as the worst case for OAL fragmentation regardless of the underlay interface IP protocol version since IPv6/IPv4 protocol translation and/or IPv6-in-IPv4 encapsulation may occur in any underlay network path. By initially assuming the IPv4 minimum even for IPv6 underlay interfaces, the OAL source may produce smaller fragments with additional encapsulation overhead but avoids loss due to presenting an underlay interface with a carrier packet that exceeds its MRU. Additionally, the OAL path could traverse multiple SRT segments with intermediate OAL forwarding nodes performing re-encapsulation where the L2 encapsulation of the previous segment is replaced by the L2 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 octets at each SRT segment for the purpose of generating OAL fragments for L2 encapsulation and transmission as carrier packets. Each successive SRT intermediate node may include either a 20 octet IPv4 or 40 octet IPv6 header, an 8 octet UDP header and in some cases an IP security encapsulation (40 octets maximum assumed) during re-encapsulation. Intermediate nodes at any SRT segment may also insert or modify the Routing Header (40 octets maximum)

following the 40 octet OAL IPv6 header and preceding the 8 octet Fragment Header. Therefore, assuming a worst case of (40 + 40 + 8) = 88 octets for L2 encapsulations plus (40 + 40 + 8) = 88 octets for OAL encapsulation leaves no less than (576 - 88 - 88) = 400 octets remaining to accommodate a portion of the original IP packet/ fragment. The OAL source therefore sets a minimum Maximum Payload Size (MPS) of 400 octets as the basis for the minimum-sized OAL fragment that can be assured of traversing all SRT 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.

The OAL source SHOULD maintain "path MPS" values for individual OAL destinations initialized to the minimum MPS and increased to larger values if better information is known or discovered. For example, when peers share a common underlay network link or a fixed path with a known larger MTU, the OAL source can set path MPS to a larger size (i.e., greater than 400 octets) as long as the peer reassembles before re-encapsulating and forwarding (while re-fragmenting if necessary). Also, if the OAL source has a way of knowing the maximum L2 encapsulation size for all SRT segments along the path it may be able to increase path MPS to reserve additional room for payload data. Even when OAL header compression is used, the OAL source must include the uncompressed OAL header size in its path MPS calculation since it may need to include a full header at any time.

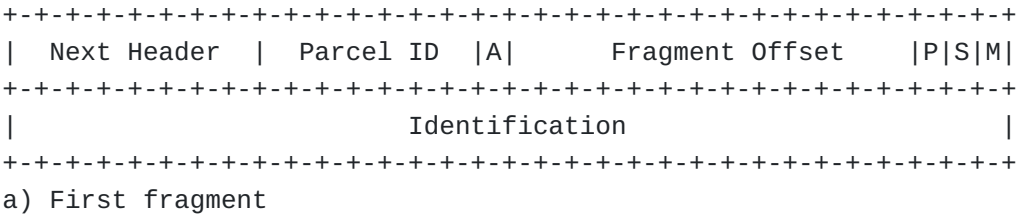
The OAL source can also optimistically set a larger path MPS and/or actively probe individual OAL destinations to discover larger sizes using packetization layer probes in a similar fashion as [RFC4821] [RFC8899], but care must be taken to avoid setting static values for dynamically changing paths leading to black holes. The probe involves sending an OAL packet larger than the current path MPS and receiving a small acknowledgement response (with the possible receipt of link-layer error message when a probe is 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) and/or with a trailing large NULL packet in a super-packet (see: Section 6.9) 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). The OAL source should maintain separate path MPS values for each (source, target) underlay interface pair for the same OAL destination, since different underlay interface pairs may support differing path MPS values.

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 at least as large

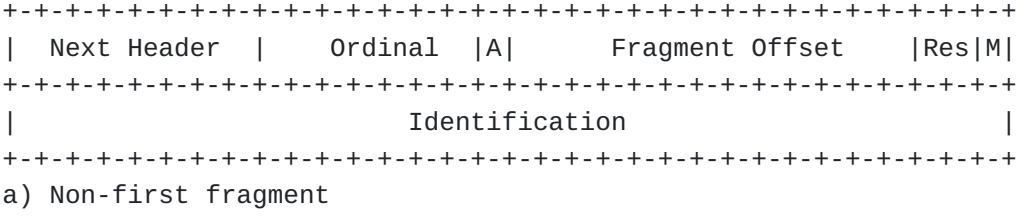
as the minimum MPS, while the final fragment MAY be smaller. The OAL source also converts all original IP packets no larger than the current MPS (or larger than 65535 octets) into atomic fragments by including a Fragment Header with Fragment Offset and More Fragments both set to 0. The OAL source then inserts a Routing Header (if necessary) following the IPv6 encapsulation header and before the Fragment Header. If the original IP packet is larger than 65535, the OAL source also inserts a Hop-By-Hop header with Jumbo Payload option immediately following the IPv6 encapsulation header and before the Routing Header (if necessary), then includes an (atomic) Fragment Header. The header extension order for each fragment therefore appears as the OAL IPv6 header followed by Hop-By-Hop header followed by Routing Header followed by Fragment Header.

The OAL source next appends the OAL checksum as the final two octets of the final fragment while increasing its (Jumbo) Payload Length by 2. If appending the checksum would cause the final fragment to exceed the current MPS, the OAL source instead reduces this "former" final fragment's Payload Length (PL) by (N*8 + (PL mod 8)) octets, where N is an integer that would result in a non-zero reduction but without causing the former final fragment to become smaller than the minimum MPS. The OAL source then creates a "new" final fragment by copying the OAL IPv6 header and extension headers from the former final fragment, then copying the (N*8 + (PL mod 8)) octets from the end of the former final fragment immediately following the new final fragment extension headers. The OAL source then sets the former final fragment's More Fragments flag to 1, increments the new final fragment's fragment offset by the former final fragment's new (PL / 8) and finally appends the checksum the same as discussed above.

Next, the OAL source replaces the IPv6 Fragment Header 1-octet "Reserved" field (and for first fragments also the 2-bit "Reserved Flags" field) with OMNI-specific encodings as shown in:



a) First fragment

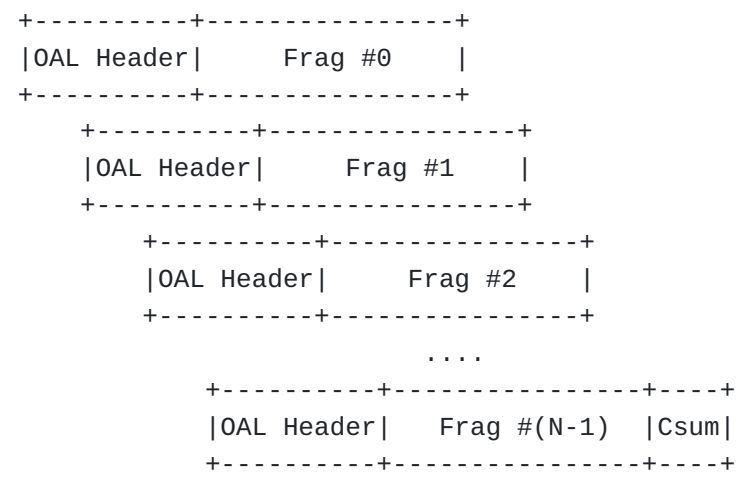


a) Non-first fragment

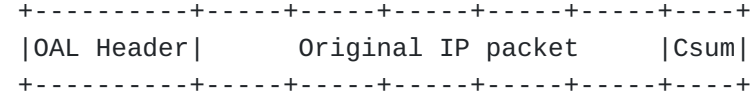
Figure 5: IPv6 Fragment Header Reserved Fields Redefined

For the first fragment, the OAL source sets the "(A)RQ" flag then sets "Parcel ID", "(P)arcel" and "(S)ub-Parcels" as specified in Section 6.14. For each non-first fragment, the OAL source instead sets the "(A)RQ" flag and writes a monotonically-increasing "Ordinal" value between 1 and 127. Specifically, the OAL source writes the ordinal number '1' for the first non-first fragment, '2' for the second, '3' for the third, etc. up to the final fragment or the ordinal value '127', whichever comes first. (For any additional non-first fragments beyond ordinal '127', the OAL source instead writes the value '0' in the Ordinal field and clears the "(A)RQ" flag. The first fragment is implicitly always considered ordinal number '0' even though the header does not include an explicit Ordinal field.)

The OAL source finally encapsulates the fragments in L2 headers to form carrier packets and forwards them over an underlay interface, while retaining the fragments and their ordinal numbers (i.e., #0, #1, #2, etc. up to #127) for a brief period to support link-layer retransmissions (see: Section 6.7). OAL fragment and carrier packet formats are shown in Figure 6.



a) OAL fragmentation (Csum in final fragment)



b) An OAL atomic fragment

c) OAL carrier packet after L2 encapsulation

Figure 6: OAL Fragments and Carrier Packets

Note: the minimum MPS assumes that any middleboxes (e.g. IPv4 NATs) that connect private networks with path MTUs smaller than 576 octets must reassemble any fragmented (outbound) IPv4 carrier packets sent by OAL sources before forwarding them to external Internetworks since middleboxes that connect OAL destinations often unconditionally drop (inbound) IPv4 fragments. However, when the path MTU in the destination private network is small, the OAL destination itself will be able to reassemble any IPv4 fragmentation that occurs in the inbound path.

6.2. OAL L2 Encapsulation and Re-Encapsulation

The OAL source or intermediate node next encapsulates each OAL fragment (with either full or compressed headers) in L2 encapsulation headers to create a carrier packet. The OAL source or intermediate node (i.e., the L2 source) includes a UDP header as the innermost sublayer if NAT traversal and/or packet filtering middlebox traversal are required; otherwise, the L2 source includes a full/compressed IP header and/or an actual link-layer header (e.g., such as for Ethernet-compatible links). The L2 source then appends any additional encapsulation sublayer headers necessary and presents the resulting carrier packet to an underlay interface, where the underlay network conveys it to a next-hop OAL intermediate node or destination (i.e., the L2 destination).

The L2 source encapsulates the OAL information immediately following the innermost L2 sublayer header. If the first four bits of the encapsulated OAL information following the innermost sublayer header encode the value '6', the information must include an uncompressed IPv6 header (plus extensions) followed by upper layer protocol headers and data. If the first four bits encode the value '4', an uncompressed IPv4 header (plus extensions) followed by upper layer protocol headers and data follows. Otherwise, the first four bits include a "Type" value, and the OAL information appears in an alternate format as specified in Section 6.4 (Types '0' and '1' are currently specified while all other values are reserved for future use). Carrier packets that contain an unrecognized Type value are unconditionally dropped.

The OAL node prepares the innermost L2 encapsulation header for OAL packets as follows:

* For UDP encapsulation, the L2 source sets the UDP source port to 8060 (i.e., the port number reserved for AERO/OMNI). When the L2 destination is a Proxy/Server or Gateway, the L2 source sets the UDP destination port to 8060; otherwise, the L2 source sets the UDP destination port to its cached port number value for the

peer. The L2 source finally sets the UDP Length the same as specified in [RFC0768]. (If the OAL packet includes an IP Jumbogram, the L2 source instead sets the UDP length to 0 and includes a Jumbo Payload option in the L2 IP header.)

*For IP encapsulation, the L2 source sets the IP {Protocol, Next-Header} to TBD1 (see: IANA Considerations) and sets the {Total, Payload} Length the same as specified in [RFC0791] or [RFC8200]. (If the OAL packet includes a true Jumbogram, the L2 source includes a Jumbo Payload option and sets {Total, Payload} Length plus the Jumbo Payload length according to the OAL length information.)

*For direct encapsulations over Ethernet-compatible links, the L2 source sets EtherType to TBD2 (see: IANA Considerations). Since the Ethernet header does not include a length field, for the OMNI EtherType the Ethernet header is followed by a four-octet Payload Length field followed immediately by the encapsulated OAL information. The Payload Length field encodes the length in octets (in network byte order) of the OAL information exclusive of the lengths of the Ethernet header and trailer.

When an L2 source includes a UDP header, it SHOULD calculate and include a UDP checksum in carrier packets with full OAL headers to prevent mis-delivery, and MAY disable UDP checksums in carrier packets with compressed OAL headers (see: Section 6.4). If the L2 source discovers that a path is dropping carrier packets with UDP checksums disabled, it should enable UDP checksums in future carrier packets sent to the same L2 destination. If the L2 source discovers that a path is dropping carrier packets that do not include a UDP header, it should include a UDP header in future carrier packets.

When an L2 source sends carrier packets with compressed OAL headers and with UDP checksums disabled, mis-delivery due to corruption of the 4-octet AERO Forwarding Vector Index (AFVI) is possible but unlikely since the corrupted index would somehow have to match valid state in the (sparsely-populated) AERO Forwarding Information Based (AFIB). In the unlikely event that a match occurs, an OAL destination may receive a mis-delivered carrier packet but can immediately reject packets with an incorrect Identification. If the Identification value is somehow accepted, the OAL destination may submit the mis-delivered carrier packet to the reassembly cache where it will most likely be rejected due to incorrect reassembly parameters. If a reassembly that includes the mis-delivered carrier packets somehow succeeds (or, for atomic fragments) the OAL destination will verify the OAL checksum to detect corruption. Finally, any spurious data that somehow eludes all prior checks will be detected and rejected by end-to-end upper layer integrity checks. See: [RFC6935][RFC6936] for further discussion.

For L2 encapsulations over IP, when the L2 source is also the OAL source it next copies the "Type of Service/Traffic Class" [RFC2983] and "Explicit Congestion Notification (ECN)" [RFC3168] values in the OAL header into the corresponding fields in the L2 IP header, then (for IPv6) set the L2 IPv6 header "Flow Label" as specified in [RFC6438]. The L2 source then sets the L2 IP TTL/Hop Limit the same as for any host (i.e., it does not copy the Hop Limit value from the OAL header) and finally sets the source and destination IP addresses to direct the carrier packet to the next hop. For carrier packets undergoing re-encapsulation, the OAL intermediate node L2 source decrements the OAL header Hop Limit and discards the carrier packet if the value reaches 0. The L2 source then copies the "Type of Service/Traffic Class" and "Explicit Congestion Notification (ECN)" values from the previous hop L2 encapsulation header into the OAL header (if present), then finally sets the source and destination IP addresses the same as above.

Following L2 encapsulation/re-encapsulation, the L2 source forwards the resulting carrier packets over one or more underlay interfaces. The underlay 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 underlay interface) to the node hosting the physical media. The OMNI interface may also apply encapsulation at the underlay 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 underlay 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 L2 Decapsulation and Reassembly

When an OMNI interface receives a carrier packet from an underlay interface, it copies the ECN value from the L2 encapsulation headers into the OAL header if the carrier packet contains a first-fragment. The OMNI interface next discards the L2 encapsulation headers and examines the OAL header of the enclosed OAL fragment. If the OAL fragment is addressed to a different node, the OMNI interface (acting as an OAL intermediate node) re-encapsulates and forwards while decrementing the OAL Hop Limit as discussed in Section 6.2. If the OAL fragment is addressed to itself, the OMNI interface (acting as an OAL destination) accepts or drops the fragment based on the (Source, Destination, Identification)-tuple and/or integrity checks.

The OAL destination next drops all non-final OAL fragments smaller than the minimum MPS and all fragments that would overlap or leave "holes" smaller than the minimum MPS with respect to other fragments already received. The OAL destination updates a checklist of accepted fragments of the same OAL packet that include an Ordinal number (i.e., Ordinals 0 through 127), but admits all accepted fragments into the reassembly cache after first removing any extension headers except for the fragment header itself. When the OAL destination receives the final fragment (i.e., the one with More Fragments set to 0), it caches the trailing checksum and reduces the Payload Length by 2. When reassembly is complete, the OAL destination verifies the OAL packet checksum and discards the packet if the checksum is incorrect. If the OAL packet was accepted, the OAL destination finally removes the OAL headers and delivers the original IP packet to the network layer.

Carrier packets often travel over paths where all links in the path include CRC-32 integrity checks for effective hop-by-hop error detection for payload sizes up to 9180 octets [CRC], but other paths may traverse links (such as tunnels over IPv4) that do not include adequate integrity protection. The OAL checksum therefore allows OAL destinations to detect reassembly misassociation splicing errors and/or carrier packet corruption caused by unprotected links [CKSUM].

The OAL checksum also provides algorithmic diversity with respect to both lower layer CRCs and upper layer Internet checksums as part of a complimentary multi-layer integrity assurance architecture. Any corruption not detected by lower layer integrity checks is therefore very likely to be detected by upper layer integrity checks that use diverse algorithms.

6.4. OAL Header Compression

OAL sources that send carrier packets with full OAL headers include a CRH-32 extension for segment-by-segment forwarding based on an AERO Forwarding Information Base (AFIB) in each OAL intermediate node. OAL source, intermediate and destination nodes can instead establish header compression state through IPv6 ND NS/NA message exchanges. After an initial NS/NA exchange, OAL nodes can apply OAL Header Compression to significantly reduce encapsulation overhead.

Each OAL node establishes AFIB soft state entries known as AERO Forwarding Vectors (AFVs) which support both carrier packet forwarding and OAL header compression/decompression. For FHS OAL sources, each AFV is referenced by a single AERO Forwarding Vector Index (AFVI) that provides compression/decompression and forwarding context for the next hop. For LHS OAL destinations, the AFV is referenced by a single AFVI that provides context for the previous

hop. For OAL intermediate nodes, the AFV is referenced by two AFVIs - one for the previous hop and one for the next hop.

When an OAL node forwards carrier packets to a next hop, it can include a full OAL header with a CRH-32 extension containing one or more AFVIs. Whenever possible, however, the OAL node should instead omit significant portions of the OAL header (including the CRH-32) while applying OAL header compression. The full or compressed OAL header follows immediately after the innermost L2 encapsulation (i.e., UDP, IP or L2) as discussed in Section 6.2. Two OAL compressed header types (Types '0' and '1') are currently specified below (note that the (A)RQ flag is always considered set and therefore omitted from the compressed headers themselves).

For OAL first-fragments (including atomic fragments), the OAL node uses OMNI Compressed Header - Type 0 (OCH-0) format as shown in Figure 7:

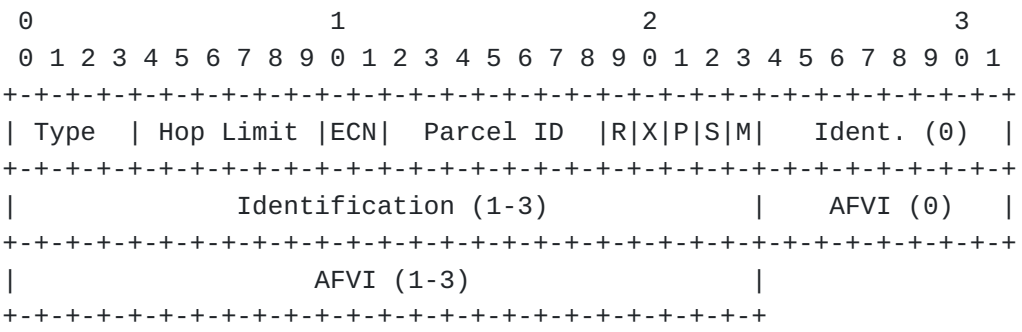


Figure 7: OMNI Compressed Header - Type 0 (OCH-0)

The format begins with a 4-bit Type, a 6-bit Hop Limit, a 2-bit Explicit Congestion Notification (ECN) field, a 7-bit Parcel ID and 5 flag bits. The format concludes with a 4-octet Identification field followed (optionally) by a 4-octet AFVI field. The OAL node sets Type to the value 0, sets Hop Limit to the minimum of the uncompressed OAL header Hop Limit and 63, sets ECN the same as for an uncompressed OAL header, and sets (Parcel ID, (P)arcel, (S)ub-parcels, (M)ore Fragments, Identification) the same as for an uncompressed fragment header. The OAL node finally sets Inde(X) and includes an AFVI if necessary; otherwise, it clears Inde(X) and omits the AFVI. (The (R)eserved flag is set to 0 on transmission and ignored on reception.)

The OAL first fragment (beginning with the original IP header) is then included immediately following the OCH-0 header, and the L2 header length field is reduced by the difference in length between the compressed headers and full-length OAL IPv6 and Fragment headers. The OAL destination can therefore determine the Payload

Length by examining the L2 header length field and/or the length field(s) in the original IP header. The OCH-0 format applies for first fragments only, which are always regarded as ordinal fragment 0 even though no explicit Ordinal field is included. The (A)RQ flag is always implicitly set, and therefore omitted from the OCH-0 header.

For OAL non-first fragments (i.e., those with non-zero Fragment Offsets), the OAL uses OMNI Compressed Header - Type 1 (OCH-1) format as shown in Figure 8:

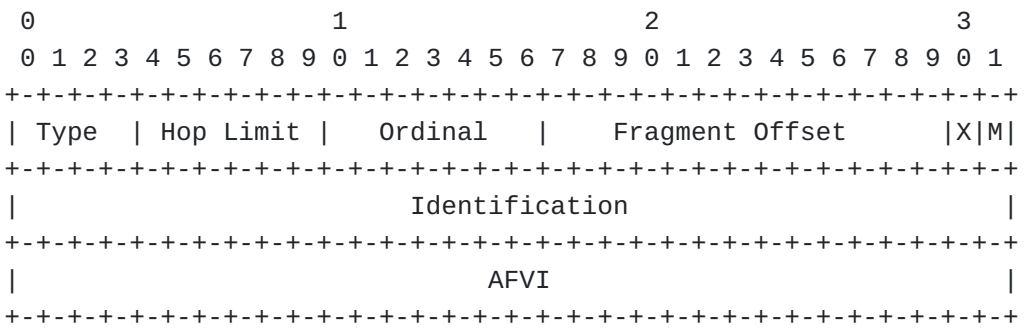


Figure 8: OMNI Compressed Header - Type 1 (OCH-1)

The format begins with a 4-bit Type, a 6-bit Hop Limit, a 7-bit Ordinal, a 13-bit Fragment Offset and 2 flag bits. The format concludes with a 4-octet Identification field followed (optionally) by a 4-octet AFVI field. The OAL node sets Type to the value 1, sets Hop Limit to the minimum of the uncompressed OAL header Hop Limit and 63, and sets (Ordinal, Fragment Offset, (M)ore Fragments, Identification) the same as for an uncompressed fragment header. If an AFVI is needed, the OAL node finally sets Inde(X) and includes an AFVI; otherwise, the node clears Inde(X) and omits the AFVI.

The OAL non-first fragment body is then included immediately following the OCH-1 header, and the L2 header length field is reduced by the difference in length between the compressed headers and full-length OAL IPv6 and Fragment headers. The OAL destination will then be able to determine the Payload Length by examining the L2 header length field. The OCH-1 format applies for non-first fragments only; therefore, the OAL source sets Ordinal to a monotonically increasing value beginning with 1 for the first non-first fragment, 2 for the second non-first fragment, etc., up to and including the final fragment. If more than 127 non-first fragments are included, these additional fragments instead set Ordinal to 0. The (A)RQ flag is always implicitly set, and therefore omitted from the OCH-1 header.

When an OAL destination or intermediate node receives a carrier packet, it determines the length of the encapsulated OAL information by examining the length field of the innermost L2 header, verifies that the innermost next header field indicates OMNI (see: Section 6.2), then examines the first four bits immediately following the innermost header. If the bits contain the value 4 or 6, the OAL node processes the remainder as an uncompressed OAL/IP header. If the bits contain a value 0 or 1, the OAL node instead processes the remainder of the header as an OCH-0/1 as specified above.

For carrier packets with OCH or full OAL headers addressed to itself and with CRH-32 extensions, the OAL node then uses the AFVI to locate the cached AFV which determines the next hop. During forwarding, the OAL node changes the AFVI to the cached value for the AFV next hop. If the OAL node is the destination, it instead reconstructs the full OAL headers then adds the resulting OAL fragment to the reassembly cache if the Identification is acceptable. (Note that for carrier packets that include an OCH-0 with both the X and M flags set to 0, the OAL node can instead locate forwarding state by examining the original IP packet header information that appears immediately after the OCH-0 header.)

Note: OAL header compression does not interfere with checksum calculation and verification, which must be applied according to the full OAL pseudo-header per Section 6.1 even when compression is used.

Note: The OCH-0/1 formats do not include the Traffic Class and Flow Label information that appears in uncompressed OAL IPv6 headers. Therefore, when OAL header compression state is initialized the Traffic Class and Flow Label are considered fixed for as long as the flow uses OCH-0/1 headers. If the flow requires frequent changes to Traffic Class and/or Flow Label information, it can include uncompressed OAL headers either continuously or periodically to update header compression state.

6.5. OAL-in-OAL Encapsulation

When an OAL source is unable to forward carrier packets directly to an OAL destination without "tunneling" through a pair of OAL intermediate nodes, the OAL source must regard the intermediate nodes as ingress and egress tunnel endpoints. This will result in nested OAL-in-OAL encapsulation in which the OAL source performs fragmentation on the inner OAL packet then forwards the fragments to the ingress tunnel endpoint which encapsulates each resulting OAL fragment in an additional OAL header before performing fragmentation following encapsulation.

For example, if the OAL source has an NCE for the OAL destination with AFVI 0x2376a7b5 and Identification 0x12345678 and the OAL ingress tunnel endpoint has an NCE for the OAL egress tunnel endpoint with AFVI 0xacdebf12 and Identification 0x98765432, the OAL source prepares the carrier packets using compressed/uncompressed OAL headers that include the AFVI and Identification corresponding to the OAL destination and with L2 header information addressed to the next hop toward the ingress tunnel endpoint. When the ingress tunnel endpoint receives the carrier packet, it recognizes the current AFVI included by the OAL source and determines the correct next hop AFVI.

The ingress tunnel endpoint then discards the L2 headers from the previous hop and encapsulates the original compressed/uncompressed OAL header within a second compressed/uncompressed OAL header while including the next-hop AFVI in the outer OAL encapsulation header and omitting the AFVI in the inner header. The ingress tunnel endpoint then includes L2 encapsulation headers with destinations appropriate for the next hop on the path to the egress tunnel endpoint. The encapsulation appears as shown in Figure 9:

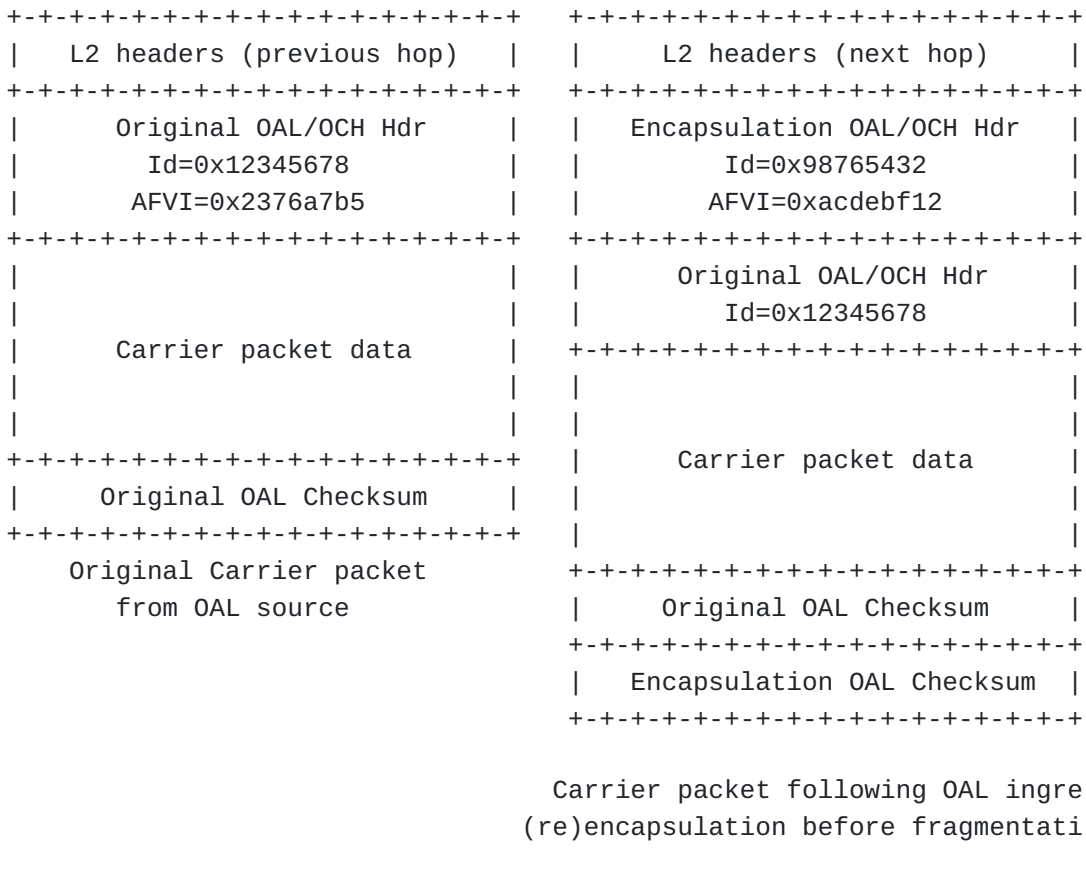


Figure 9: Carrier Packet in Carrier Packet Encapsulation

Note that only a single OAL-in-OAL encapsulation layer is supported, and that AFVIs appear only in the outer OAL header (i.e., either within a CRH-32 routing header when a full OAL header is used or within an OCH header with X set to 0). The inner OAL header should omit the CRH-32 header or use an OCH header with X set to 1, respectively.

Note that OAL/OCH encapsulation may cause the payloads of OAL packets produced by the ingress tunnel endpoint to exceed the minimum MPS by a small amount. If the ingress has assurance that the path to the egress will include only links capable of transiting the resulting (slightly larger) carrier packets it should forward without further fragmentation. Otherwise, the ingress must perform fragmentation following encapsulation to produce two fragments such that the size of the first fragment matches the size of the original OAL packet, and with the remainder in a second fragment. The egress tunnel endpoint must then reassemble then decapsulate to arrive at the original OAL packet which is then subject to further forwarding.

6.6. OAL Identification Window Maintenance

The OAL encapsulates each original IP packet as an OAL packet then performs fragmentation to produce one or more carrier packets with the same 32-bit Identification value. In environments where spoofing is not considered a threat, OMNI interfaces send OAL packets with Identifications beginning with an unpredictable Initial Send Sequence (ISS) value [RFC7739] monotonically incremented (modulo 2**32) for each successive OAL packet sent to either a specific neighbor or to any neighbor. (The OMNI interface may later change to a new unpredictable ISS value as long as the Identifications are assured unique within a timeframe that would prevent the fragments of a first OAL packet from becoming associated with the reassembly of a second OAL packet.) In other environments, OMNI interfaces should maintain explicit per-neighbor send and receive windows to detect and exclude spurious carrier packets that might clutter the reassembly cache as discussed below.

OMNI interface neighbors use TCP-like synchronization to maintain windows with unpredictable ISS values incremented (modulo 2**32) for each successive OAL packet and re-negotiate windows often enough to maintain an unpredictable profile. OMNI interface neighbors exchange IPv6 ND messages with OMNI options that include TCP-like information fields to manage streams of OAL packets instead of streams of octets. As a link-layer service, the OAL provides low-persistence best-effort retransmission with no mitigations for duplication, reordering or deterministic delivery. Since the service model is best-effort and only control message sequence numbers are acknowledged, OAL nodes can select unpredictable new initial

sequence numbers outside of the current window without delaying for the Maximum Segment Lifetime (MSL).

OMNI interface neighbors maintain current and previous window state in IPv6 ND NCEs to support dynamic rollover to a new window while still sending OAL packets and accepting carrier packets from the previous windows. Each NCE is indexed by the neighbor's ULA, while the OAL encapsulation ULA (which may be different) provides context for Identification verification. OMNI interface neighbors synchronize windows through asymmetric and/or symmetric IPv6 ND message exchanges. When a node receives an IPv6 ND message with new window information, it resets the previous window state based on the current window then resets the current window based on new and/or pending information.

The IPv6 ND message OMNI option header extension sub-option includes TCP-like information fields including Sequence Number, Acknowledgement Number, Window and flags (see: Section 12). OMNI interface neighbors maintain the following TCP-like state variables in the NCE:

Send Sequence Variables (current, previous and pending)

```
   SND.NXT - send next
   SND.WND - send window
   ISS     - initial send sequence number
```

Receive Sequence Variables (current and previous)

```
   RCV.NXT - receive next
   RCV.WND - receive window
   IRS     - initial receive sequence number
```

OMNI interface neighbors "OAL A" and "OAL B" exchange IPv6 ND messages per [RFC4861] with OMNI options that include TCP-like information fields. When OAL A synchronizes with OAL B, it maintains both a current and previous SND.WND beginning with a new unpredictable ISS and monotonically increments SND.NXT for each successive OAL packet transmission. OAL A initiates synchronization by including the new ISS in the Sequence Number of an authentic IPv6 ND message with the SYN flag set and with Window set to M (up to 2**24) as a tentative receive window size while creating a NCE in the INCOMPLETE state if necessary. OAL A caches the new ISS as pending, uses the new ISS as the Identification for OAL encapsulation, then sends the resulting OAL packet to OAL B and waits up to RetransTimer milliseconds to receive an IPv6 ND message response with the ACK flag set (retransmitting up to MAX_UNICAST_SOLICIT times if necessary).

When OAL B receives the SYN, it creates a NCE in the STALE state if necessary, resets its RCV variables, caches the tentative (send) window size M, and selects a (receive) window size N (up to 2**24) to indicate the number of OAL packets it is willing to accept under the current RCV.WND. (The RCV.WND should be large enough to minimize control message overhead yet small enough to provide an effective filter for spurious carrier packets.) OAL B then prepares an IPv6 ND message with the ACK flag set, with the Acknowledgement Number set to OAL A's next sequence number, and with Window set to N. Since OAL B does not assert an ISS of its own, it uses the IRS it has cached for OAL A as the Identification for OAL encapsulation then sends the ACK to OAL A.

When OAL A receives the ACK, it notes that the Identification in the OAL header matches its pending ISS. OAL A then sets the NCE state to REACHABLE and resets its SND variables based on the Window size and Acknowledgement Number (which must include the sequence number following the pending ISS). OAL A can then begin sending OAL packets to OAL B with Identification values within the (new) current SND.WND for up to ReachableTime milliseconds or until the NCE is updated by a new IPv6 ND message exchange. This implies that OAL A must send a new SYN before sending more than N OAL packets within the current SND.WND, i.e., even if ReachableTime is not nearing expiration. After OAL B returns the ACK, it accepts carrier packets received from OAL A within either the current or previous RCV.WND as well as any new authentic NS/RS SYN messages received from OAL A even if outside the windows.

OMNI interface neighbors can employ asymmetric window synchronization as described above using two independent (SYN -> ACK) exchanges (i.e., a four-message exchange), or they can employ symmetric window synchronization using a modified version of the TCP three-way handshake as follows:

* OAL A prepares a SYN with an unpredictable ISS not within the current SND.WND and with Window set to M as a tentative receive window size. OAL A caches the new ISS and Window size as pending information, uses the pending ISS as the Identification for OAL encapsulation, then sends the resulting OAL packet to OAL B and waits up to RetransTimer milliseconds to receive an ACK response (retransmitting up to MAX_UNICAST_SOLICIT times if necessary).

* OAL B receives the SYN, then resets its RCV variables based on the Sequence Number while caching OAL A's tentative receive Window size M and a new unpredictable ISS outside of its current window as pending information. OAL B then prepares a response with Sequence Number set to the pending ISS and Acknowledgement Number set to OAL A's next sequence number. OAL B then sets both the SYN and ACK flags, sets Window to N and sets the OPT flag

according to whether an explicit concluding ACK is optional or mandatory. OAL B then uses the pending ISS as the Identification for OAL encapsulation, sends the resulting OAL packet to OAL A and waits up to RetransTimer milliseconds to receive an acknowledgement (retransmitting up to MAX_UNICAST_SOLICIT times if necessary).

*OAL A receives the SYN/ACK, then resets its SND variables based on the Acknowledgement Number (which must include the sequence number following the pending ISS) and OAL B's advertised Window N. OAL A then resets its RCV variables based on the Sequence Number and marks the NCE as REACHABLE. If the OPT flag is clear, OAL A next prepares an immediate solicited NA message with the ACK flag set, the Acknowledgement Number set to OAL B's next sequence number, with Window set a value that may be the same as or different than M, and with the OAL encapsulation Identification to SND.NXT, then sends the resulting OAL packet to OAL B. If the OPT flag is set and OAL A has OAL packets queued to send to OAL B, it can optionally begin sending their carrier packets under the (new) current SND.WND as implicit acknowledgements instead of returning an explicit ACK. In that case, the tentative Window size M becomes the current receive window size.

*OAL B receives the implicit/explicit acknowledgement(s) then resets its SND state based on the pending/advertised values and marks the NCE as REACHABLE. If OAL B receives an explicit acknowledgement, it uses the advertised Window size and abandons the tentative size. (Note that OAL B sets the OPT flag in the SYN/ACK to assert that it will interpret timely receipt of carrier packets within the (new) current window as an implicit acknowledgement. Potential benefits include reduced delays and control message overhead, but use case analysis is outside the scope of this specification.)

Following synchronization, OAL A and OAL B hold updated NCEs and can exchange OAL packets with Identifications set to SND.NXT while the state remains REACHABLE and there is available window capacity. Either neighbor may at any time send a new SYN to assert a new ISS. For example, if OAL A's current SND.WND for OAL B is nearing exhaustion and/or ReachableTime is nearing expiration, OAL A continues to send OAL packets under the current SND.WND while also sending a SYN with a new unpredictable ISS. When OAL B receives the SYN, it resets its RCV variables and may optionally return either an asymmetric ACK or a symmetric SYN/ACK to also assert a new ISS. While sending SYNs, both neighbors continue to send OAL packets with Identifications set to the current SND.NXT then reset the SND variables after an acknowledgement is received.

While the optimal symmetric exchange is efficient, anomalous conditions such as receipt of old duplicate SYNs can cause confusion for the algorithm as discussed in Section 3.4 of [RFC0793]. For this reason, the OMNI option header includes an RST flag which OAL nodes set in solicited NA responses to ACKs received with incorrect acknowledgement numbers. The RST procedures (and subsequent synchronization recovery) are conducted exactly as specified in [RFC0793].

OMNI interfaces may set the PNG ("ping") flag when a reachability confirmation outside the context of the IPv6 ND protocol is needed (OMNI interfaces therefore most often set the PNG flag in advertisement messages and ignore it in solicitation messages). When an OMNI interface receives a PNG, it returns an unsolicited NA (uNA) ACK with the PNG message Identification in the Acknowledgment, but without updating RCV state variables. OMNI interfaces return unicast uNA ACKs even for multicast PNG destination addresses, since OMNI link multicast is based on unicast emulation.

OMNI interfaces that employ the window synchronization procedures described above observe the following requirements:

* OMNI interfaces MUST select new unpredictable ISS values that are at least a full window outside of the current SND.WND.

* OMNI interfaces MUST set the initial SYN message Window field to a tentative value to be used only if no concluding NA ACK is sent.

* OMNI interfaces that receive advertisements with the PNG and/or SYN flag set MUST NOT set the PNG and/or SYN flag in uNA responses.

* OMNI interfaces that send advertisements with the PNG and/or SYN flag set MUST ignore uNA responses with the PNG and/or SYN flag set.

* OMNI interfaces MUST send IPv6 ND messages used for window synchronization securely while using unpredictable initial Identification values until synchronization is complete.

Note: Although OMNI interfaces employ TCP-like window synchronization and support uNA ACK responses to SYNs and PNGs, all other aspects of the IPv6 ND protocol (e.g., control message exchanges, NCE state management, timers, retransmission limits, etc.) are honored exactly per [RFC4861].

Note: Recipients of OAL-encapsulated IPv6 ND messages index the NCE based on the message source address, which also determines the carrier packet Identification window. However, IPv6 ND messages may

contain a message source address that does not match the OMNI encapsulation source address when the recipient acts as a proxy.

Note: OMNI interface neighbors apply the same send and receive windows for all of their (multilink) underlay interface pairs that exchange carrier packets. Each interface pair represents a distinct underlay network path, and the set of paths traversed may be highly diverse when multiple interface pairs are used. OMNI intermediate nodes therefore SHOULD NOT cache window synchronization parameters in IPv6 ND messages they forward since there is no way to ensure network-wide middlebox state consistency.

6.7. OAL Fragment Retransmission

When the OAL source sends carrier packets to an OAL destination, it should cache recently sent packets in case timely best-effort selective retransmission is requested. The OAL destination in turn maintains a checklist for the (Source, Destination, Identification)-tuple of recently received carrier packets and notes the ordinal numbers of OAL packet fragments already received (i.e., as Frag #0, Frag #1, Frag #2, etc.). The timeframe for maintaining the OAL source and destination caches determines the link persistence (see: [RFC3366]).

If the OAL destination notices some fragments missing after most other fragments within the same link persistence timeframe have already arrived, it may issue an Automatic Repeat Request (ARQ) with Selective Repeat (SR) by sending a uNA message to the OAL source. The OAL destination creates a uNA message with an OMNI option with one or more Fragmentation Report (FRAGREP) sub-options that include a list of (Identification, Bitmap)-tuples for fragments received and missing from this OAL source (see: Section 12 and [I-D.templin-6man-fragrep]). The OAL destination includes an authentication signature if necessary, 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 then examines the FRAGREP. For each (Source, Destination, Identification)-tuple, the OAL source determines whether it still holds the corresponding carrier packets in its cache and retransmits any for which the Bitmap indicates a loss event. For example, if the Bitmap indicates that ordinal fragments #3, #7, #10 and #13 from the OAL packet with Identification 0x12345678 are missing the OAL source only retransmits carrier packets containing those fragments. When the OAL destination receives the retransmitted carrier packets, it admits the enclosed fragments into the reassembly cache and updates its checklist. If some fragments are still missing, the OAL

destination may send a small number of additional uNA ARQ/SRs within the link persistence timeframe.

The OAL therefore provides a link-layer low-to-medium persistence ARQ/SR service consistent with [RFC3366] and Section 8.1 of [RFC3819]. The service provides the benefit of timely best-effort link-layer retransmissions which may reduce packet loss and avoid some unnecessary end-to-end delays. This best-effort network-based service therefore compliments higher layer end-to-end protocols responsible for true reliability.

6.8. 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 and also in [I-D.templin-6man-fragrep].

Ordinary PTB messages with ICMPv4 header "unused" field or ICMPv6 header Code field value 0 are hard errors that always indicate that a packet has been dropped due to a real MTU restriction. However, the 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, number of fragments needed, 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 the packet was dropped or 2 if the packet was forwarded successfully. 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 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 first fragment 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.)

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) that include OMNI encapsulated PTB messages with code 1 or 2. The OAL destination creates a uNA message with an OMNI option containing an authentication message sub-option if necessary followed optionally by a ICMPv6 Error sub-option that encodes a PTB message with a reduced value and with 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 the leading portion of the OAL first fragment (beginning with the OAL header) in the PTB "packet in error" field, 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 sends a corresponding network layer PTB soft error to the original source to recommend a smaller size. The OAL source crafts the PTB by extracting the leading portion of the original IP packet from the OMNI encapsulated PTB message (i.e., not including the OAL header) and writes it in the "packet in error" field of a network layer PTB with destination set to the original IP packet source and source set to one of its OMNI interface addresses that is routable from the perspective of the original source.

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 resume 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.

6.9. OAL Super-Packets

By default, the OAL source includes a 40-octet IPv6 encapsulation header for each original IP packet during OAL encapsulation. The OAL source also calculates then performs fragmentation such that a copy of the 40-octet IPv6 header plus an 8-octet IPv6 Fragment Header is included in each OAL fragment (when a Routing Header 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]) supports encapsulation of multiple original IP packets and/or control messages 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. 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 a trailing checksum field included in the final fragment. The OAL super-packet format is transposed from [I-D.ietf-intarea-tunnels] and shown in Figure 10:

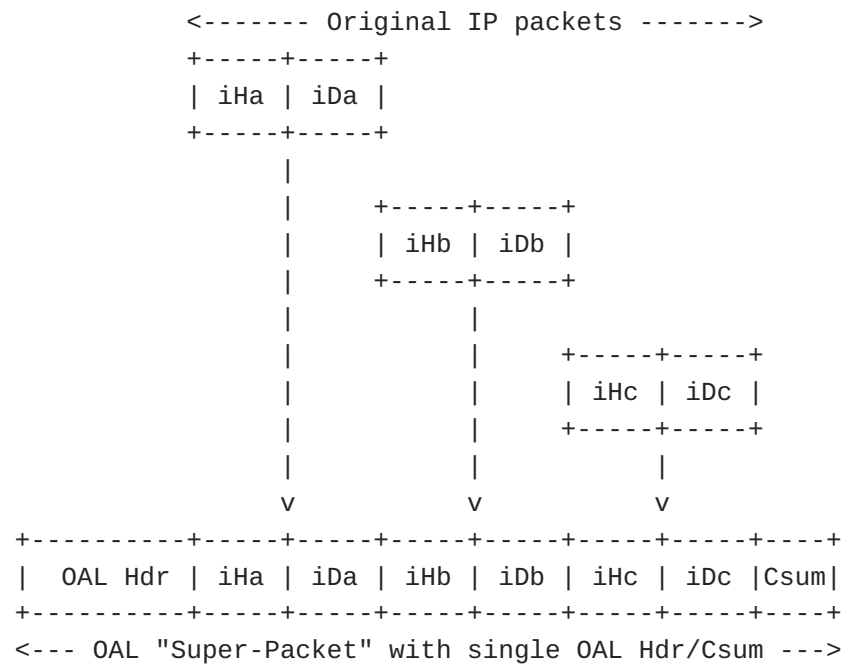


Figure 10: OAL Super-Packet Format

When the OAL source prepares a super-packet, it applies OAL fragmentation, includes a trailing checksum in the final fragment, applies L2 encapsulation to each fragment then sends the resulting carrier packets to the OAL destination. When the OAL destination receives the super-packet it sets aside the trailing checksum, reassembles if necessary, then verifies the checksum while regarding 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.

When an OAL source prepares a super-packet that includes an IPv6 ND message with an authentication signature or ICMPv6 message checksum as the first original IP packet (i.e., iHa/iDa), it calculates the authentication signature or checksum over the remainder of super-packet. Security and integrity for forwarding initial protocol data packets in conjunction with IPv6 ND messages used to establish NCE state are therefore supported. (A common use case entails a path MPS probe beginning with a signed IPv6 ND message followed by a NULL IPv6 packet with a suitably large (Jumbo) Payload Length but with Next Header set to 59 for "No Next Header".)

6.10. OAL Bubbles

OAL sources may send NULL OAL packets known as "bubbles" for the purpose of establishing Network Address Translator (NAT) state on the path to the OAL destination. The OAL source prepares a bubble by crafting an OAL header with appropriate IPv6 source and destination ULAs, with the IPv6 Next Header field set to the value 59 ("No Next Header" - see [RFC8200]) and with only the trailing OAL Checksum field (i.e., and no protocol data) immediately following the IPv6 header.

The OAL source includes a random Identification value then encapsulates the OAL packet in L2 headers destined to either the mapped address of the OAL destination's first-hop ingress NAT or the L2 address of the OAL destination itself. When the OAL source sends the resulting carrier packet, any egress NATs in the path toward the L2 destination will establish state based on the activity but the bubble will be harmlessly discarded by either an ingress NAT on the path to the OAL destination or by the OAL destination itself.

The bubble concept for establishing NAT state originated in [RFC4380] and was later updated by [RFC6081]. OAL bubbles may be employed by mobility services such as [I-D.templin-6man-aero].

6.11. OAL Requirements

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

* OAL sources MUST forward original IP packets either larger than the OMNI interface MRU or smaller than the minimum MPS minus the trailing checksum size as atomic fragments (i.e., and not as multiple fragments).

* OAL sources MUST produce non-final fragments with payloads no smaller than the minimum MPS during fragmentation.

* OAL intermediate nodes SHOULD and OAL destinations MUST unconditionally drop any non-final OAL fragments with payloads smaller than the minimum MPS.

* OAL destinations MUST drop any new OAL fragments with offset and length that would overlap with other fragments and/or leave holes smaller than the minimum MPS between fragments that have already been received.

Note: Under the minimum MPS, ordinary 1500 octet original IP packets would require at most 4 OAL fragments, with each non-final fragment containing 400 payload octets and the final fragment containing 302 payload octets (i.e., the final 300 octets of the original IP packet

plus the 2 octet trailing checksum). 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 underlay interface pair instead of spread across multiple underlay interface pairs. 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 ordinary hardware platforms. Even so, while the OAL destination is reporting reassembly congestion (see: Section 6.8) the OAL source could impose "pacing" by inserting an inter-fragment delay and increasing or decreasing the delay according to congestion indications.

6.12. 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 underlay interface such that congestion experienced over a first underlay interface does not cause discard of incomplete reassemblies for uncongested underlay interfaces.

3. Attacks based on predictable fragment identification values - in environments where spoofing is possible, this threat is mitigated through the use of Identification windows beginning with unpredictable values per Section 6.6. By maintaining windows of acceptable Identifications, OAL neighbors can quickly discard spurious carrier packets that might otherwise

clutter the reassembly cache. The OAL additionally provides an integrity check to detect corruption that may be caused by spurious fragments received with in-window Identification values.

4. Evasion of Network Intrusion Detection Systems (NIDS) - since the OAL source employs a robust MPS, network-based firewalls can inspect and drop OAL fragments containing malicious data thereby disabling reassembly by the OAL destination. However, since OAL fragments may take different paths through the network (some of which may not employ a firewall) each OAL destination must also employ a firewall.

IPv4 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 IP ID are still alive in the network [RFC4963]. Since carrier packets sent via an IPv4 path with DF=0 are normally no larger than 576 octets, IPv4 fragmentation is possible only at small-MTU links in the path which should support data rates low enough for safe reassembly [RFC3819]. (IPv4 carrier packets larger than 576 octets with DF=0 may incur high data rate reassembly errors in the path, but the OAL checksum provides OAL destination integrity assurance.) Since IPv6 provides a 32-bit Identification value, IP ID wraparound at high data rates is not a concern for IPv6 fragmentation.

Fragmentation security concerns for large IPv6 ND messages are documented in [RFC6980]. 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 minimum/path MPS for this OAL destination.

Unless the path is secured at the network-layer or below (i.e., in environments where spoofing is possible), OMNI interfaces MUST NOT send ordinary carrier packets with Identification values outside the current window and MUST secure IPv6 ND messages used for address resolution or window state synchronization. OAL destinations SHOULD therefore discard without reassembling any out-of-window OAL fragments received over an unsecured path.

6.13. OMNI Hosts

OMNI Hosts are end systems that extend the OMNI link over ENET underlay interfaces (i.e., either as an OMNI interface or as a sublayer of the ENET interface itself). Each ENET is connected to the rest of the OMNI link by a Client that receives an MNP

delegation. Clients delegate MNP addresses and/or sub-prefixes to ENET nodes (i.e., Hosts, other Clients, routers and non-OMNI hosts) using standard mechanisms such as DHCP [RFC8415][RFC2131] and IPv6 Stateless Address AutoConfiguration (SLAAC) [RFC4862]. Clients forward packets between their ENET Hosts and peers on external networks acting as routers and/or OAL intermediate nodes.

OMNI Hosts coordinate with Clients and/or other Hosts connected to the same ENET using IP-encapsulated IPv6 ND messages. The IP encapsulation headers and ND messages both use the MNP-based addresses assigned to ENET underlay interfaces as source and destination addresses (i.e., instead of ULAs). For IPv4 MNPs, the ND messages use IPv4-Compatible IPv6 addresses [RFC4291] in place of the IPv4 addresses. (Note that IPv4-Compatible IPv6 addresses are deprecated for all other uses by the aforementioned standard.)

Hosts discover Clients by sending encapsulated RS messages using an OMNI link IP anycast address (or the unicast address of the Client) as the RS L2 encapsulation destination as specified in Section 15. The Client configures the IPv4 and/or IPv6 anycast addresses for the OMNI link on its ENET interface and advertises the address(es) into the ENET routing system. The Client then responds to the encapsulated RS messages by sending an encapsulated RA message that uses its ENET unicast address as the source. (To differentiate itself from an INET border Proxy/Server, the Client sets the RA message OMNI Interface Attributes sub-option LHS field to 0 for the Host's interface index. When the RS message includes an L2 anycast destination address, the Client also includes an Interface Attributes sub-option for interface index 0 to inform the Host of its L2 unicast address - see: Section 15 for full details on the RS and RA message contents.)

Hosts coordinate with peer Hosts on the same ENET by sending encapsulated NS messages to receive an NA reply. (Hosts determine whether a peer is on the same ENET by matching the peer's IP address with the MNP (sub)-prefix for the ENET advertised in the Client's RA message [RFC8028].) Each ENET peer then creates a NCE and synchronizes Identification windows the same as for OMNI link neighbors, and the Host can then engage in OMNI link transactions with the Client and/or other ENET Hosts. By coordinating with the Client in this way, the Host treats the Client as if it were an ANET Proxy/Server, and the Client provides the same services that a Proxy/Server would provide. By coordinating with other Hosts, the peer hosts can exchange large IP packets or parcels over the ENET using IPv6 fragmentation if necessary.

When a Host prepares an IP packet or parcel, it uses the IP address of its native ENET interface as the source and the IP address of the (remote) peer as the destination. The Host next performs parcel

segmentation if necessary (see: Section 6.14) then encapsulates the packet/parcel in an IP header of the version supported by the ENET while setting the source to the same address and destination to either the same address if the peer is on the local ENET, or to the IP address of the Client otherwise. The Host can then proceed to exchange packets/parcels with the destination, either directly or via the Client as an intermediate node.

The encapsulation procedures are coordinated per Section 6.1, except that the IP encapsulation header version matches the native ENET IP protocol version and uses IPv6 GUA or public/private IPv4 addresses instead of ULAs. The Host sets the encapsulation IP header {Protocol, Next-Header} field to TBD1 to indicate that this is an OAL encapsulation and not an ordinary IP-in-IP encapsulation. When the inner header is IPv4-based, the Host next translates the encapsulation header into an IPv6 header with IPv4-Compatible addresses while setting the [IPv6 Traffic Class, Payload Length, Next Header, Hop Limit] fields according to the IPv4 {Type of Service, Total Length, Protocol, TTL} fields, respectively, while setting Flow Label to 0. The Host then calculates an OAL checksum, writes the value as the final two octets of the encapsulated packet then applies IPv6 fragmentation to the encapsulated packet to produce IPv6 fragments no smaller than the MPS the same as described in Section 6.1. If the original encapsulation IP header was IPv4, the Host next translates the IPv6 encapsulation headers back to IPv4 headers with Protocol value set to 44 since the immediately next header is the IPv6 Fragment Header. The Host finally sends the IP encapsulated fragments to the ENET peer.

When the ENET peer receives IP encapsulated fragments, for IPv4 it first translates the encapsulation headers back to IPv6 headers with IPv4-Compatible addresses the same as above. The peer then reassembles and verifies the OAL checksum. If the checksum is correct, the peer next removes the encapsulation headers and applies parcel reassembly if necessary. The peer then either delivers the encapsulated packet/parcel to upper layers if the peer is the destination or forwards the packet/parcel toward the final destination if the peer is a Client acting as an intermediate node.

Hosts and Clients that initiate OMNI-based packet/parcel transactions should first test the path toward the final destination using the parcel path qualification procedure specified in [I-D.templin-intarea-parcels]. An OMNI Host that sends and receives parcels need not implement the full OMNI interface abstraction but MUST implement enough of the OAL to be capable of fragmenting and reassembling maximum-length encapsulated IP packets/parcels and sub-parcels as discussed above and in the following section.

6.14. IP Parcels

IP parcels are specified in [I-D.templin-intarea-parcels], while details for their application over OMNI interfaces is specified here. IP parcels are formed by an OMNI Host or Client upper layer protocol entity (identified by the "5-tuple" source IP address/port number, destination IP address/port number and protocol number) when it produces a protocol data unit containing the concatenation of up to 64 upper layer protocol segments. All non-final segments MUST be equal in length while the final segment MUST NOT be larger but MAY be smaller. Each non-final segment MUST be no larger than 65535 minus the length of the IP header plus extensions, minus the length of the OAL encapsulation header and trailer. The upper layer protocol then presents the buffer and non-final segment size to the IP layer which appends a single IP header (plus any extension headers) before presenting the parcel to the OMNI Interface.

For IPv4, the IP layer prepares the parcel by appending an IPv4 header with a Jumbo Payload option (see: Section 5.1) where "Jumbo Payload Length" is a 32-bit unsigned integer value (in network byte order) set to the lengths of the IPv4 header plus all concatenated segments. The IP layer next sets the IPv4 header DF bit to 1, then sets the IPv4 header Total Length field to the length of the IPv4 header plus the length of the first segment only. (Note: the IP layer can form true IPv4 jumbograms (as opposed to parcels) by instead setting the Total Length field to the length of the IPv4 header only.)

For IPv6, the IP layer forms a parcel by appending an IPv6 header with a Jumbo Payload option the same as for IPv4 above where "Jumbo Payload Length" is set to the lengths of the IPv6 Hop-by-Hop Options header and any other extension headers present plus all concatenated segments. The IP layer next sets the IPv6 header Payload Length field to the lengths of the IPv6 Hop-by-Hop Options header and any other extension headers present plus the length of the first segment only. (Note: the IP layer can form true IPv6 jumbograms (as opposed to parcels) by instead setting the Payload Length field to 0.)

An IP parcel therefore has the following structure:

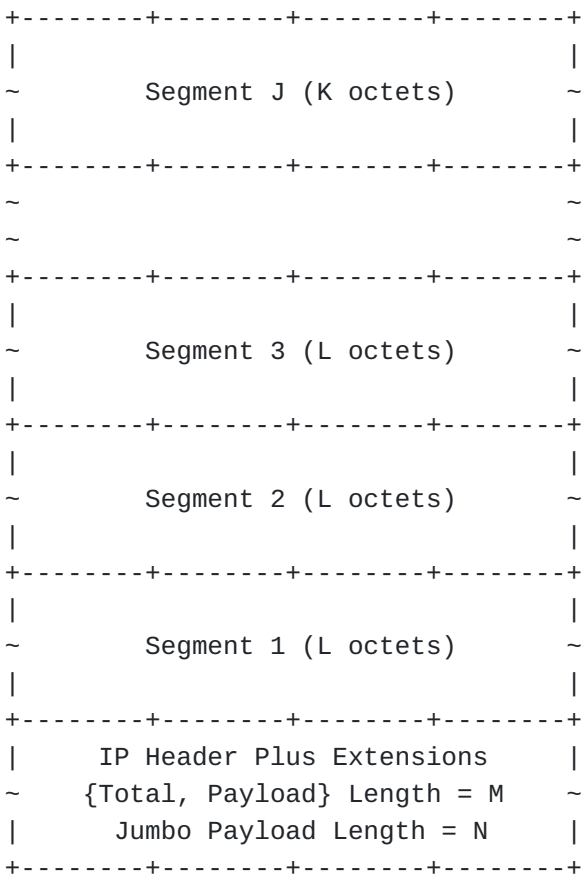


Figure 11: OMNI Interface IP Parcels

where J is the total number of segments (between 1 and 64), L is the length of each non-final segment which MUST NOT be larger than 65535 (minus headers as above) and K is the length of the final segment which MUST NOT be larger than L. The values M and N are then set to the length of the IP header plus extensions for IPv4 or to the length of the extensions only for IPv6, then further calculated as follows:

```
M = M + ((J-1) ? L : K)

N = N + (((J-1) * L) + K)
```

Note: a "singleton" parcel is one that includes only the IP header plus extensions with a single segment of length K, while a "null" parcel is a singleton with K=0, i.e., a parcel consisting of only the IP header plus extensions with no octets beyond.

When the IP layer forwards a parcel, the OMNI interface invokes the OAL which forwards it to either a Client as an intermediate node or the final destination itself. The OAL source first assigns a monotonically-incrementing (modulo 127) "Parcel ID" and subdivides the parcel into sub-parcels no larger than the maximum of the path

MTU to the next hop or 64KB (minus the length of encapsulation headers). The OAL source determines the number of segments of length L that can fit into each sub-parcel under these size constraints, e.g. if the OAL source determines that a sub-parcel can contain 3 segments of length L, it creates sub-parcels with the first containing segments 1-3, the second containing segments 4-6, etc. and with the final containing any remaining segments. The OAL source then appends an identical IP header plus extensions to each sub-parcel while resetting M and N in each according to the above equations with J set to 3 and K set to L for each non-final sub-parcel and with J set to the remaining number of segments for the final sub-parcel.

The OAL source next performs encapsulation on each sub-parcel with destination set to the next hop address. If the next hop is reached via an ANET/INET interface, the OAL source inserts an OAL header the same as discussed in Section 6.1 and sets the destination to the ULA-MNP of the target Client. If the next hop is reached via an ENET interface, the OAL source instead inserts an IP header of the appropriate protocol version for the underlay ENET (i.e., even if the encapsulation header is IPv4) and sets the destination to the ENET IP address of the next hop. The OAL source inserts the encapsulation header even if no actual fragmentation is needed and/or even if the Jumbo Payload option is present.

The OAL source next assigns an Identification number that is monotonically-incremented for each consecutive sub-parcel, calculates and appends the OAL checksum, then performs IPv6 fragmentation over the sub-parcel if necessary to create fragments small enough to traverse the path to the next hop. (If the encapsulation header is IPv4, the OAL source first translates the encapsulation header into an IPv6 header with IPv4-Compatible IPv6 addresses before performing the fragmentation/reassembly operation while inserting an IPv6 Fragment Header.) The OAL source then writes the "Parcel ID" and sets/clears the "(P)arcel" and "(More) (S)ub-Parcels" bits in the Fragment Header of the first fragment (see: Figure 5). (The OAL source sets P to 1 for a parcel or to 0 for a non-parcel. When P is 1, the OAL next sets S to 1 for non-final sub-parcels or to 0 if the sub-parcel contains the final segment.) The OAL source then forwards each IP encapsulated packet/fragment to the next hop (i.e., after first translating the IPv6 encapsulation header back to IPv4 if necessary).

When the next hop receives the encapsulated IP fragments or whole packets, it acts as an OAL destination and reassembles if necessary (i.e., after first translating the IPv4 encapsulation header to IPv6 if necessary). If the P flag in the first fragment is 0, the OAL destination then processes the reassembled entity as an ordinary IP packet; otherwise it continues processing as a sub-parcel. If the

OAL destination is not the final destination, it retains the sub-parcels along with their Parcel ID and Identification values for a brief time in hopes of re-combining with peer sub-parcels of the same original parcel identified by the 4-tuple consisting of the IP encapsulation source and destination, Identification and Parcel ID. The OAL destination re-combines peers by concatenating the segments included in sub-parcels with the same Parcel ID and with Identification values within 64 of one another to create a larger sub-parcel possibly even as large as the entire original parcel. Order of concatenation is not important, with the exception that the final sub-parcel (i.e., the one with S set to 0) must occur as the final concatenation before transmission. The OAL destination then appends a common IP header plus extensions to each re-combined sub-parcel while resetting M and N in each according to the above equations with J, K and L set accordingly.

When the current OAL destination is an intermediate node, it next becomes an OAL source to forward the re-combined (sub-)parcel(s) to the next hop toward the final destination using encapsulation/translation the same as specified above. (Each such intermediate node MUST ensure that the S flag remains set to 0 in the sub-parcel that contains the final segment.) When the parcel or sub-parcels arrive at the final OAL destination, it re-combines them into the largest possible (sub)-parcels while honoring the S flag then delivers them to upper layers which act on the enclosed 5-tuple information supplied by the original source.

Note: while the final destination may be tempted to re-combine the sub-parcels of multiple different parcels with identical upper layer protocol 5-tuples and with non-final segments of identical length, this process could become complicated when the different parcels each have final segments of diverse lengths. Since this could possibly defeat any perceived performance advantages, the decision of whether and how to perform inter-parcel concatenation is an implementation matter.

7. Frame Format

When the OMNI interface forwards original IP packets from the network layer it first invokes the OAL to create OAL packets/fragments if necessary, then includes any L2 encapsulations and finally engages the native frame format of the underlay 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 L2 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 underlay interface.

8. Link-Local Addresses (LLAs)

[RFC4861] requires that nodes assign Link-Local Addresses (LLAs) to all interfaces, and that routers use their LLAs as the source address for RA and Redirect messages. OMNI interfaces honor the first requirement, but do not honor the second since the OMNI link could consist of the concatenation of multiple links with diverse ULA prefixes (see Section 9) but for which multiple nodes might configure identical interface identifiers (IIDs). OMNI interface LLAs are therefore considered only as context for IID formation as discussed below and have no other operational role.

OMNI interfaces assign IPv6 LLAs through pre-service administrative actions. Clients assign "LLA-MNPs" with IIDs that embed the Client's unique MNP, while Proxy/Servers assign "LLA-RNDs" that include a randomly-generated IIDs generated as specified in [RFC7217]. LLAs are configured as follows:

* IPv6 LLA-MNPs encode the most-significant 64 bits of an MNP within the least-significant 64 bits of the IPv6 link-local prefix fe80::/64, i.e., in the IID portion of the LLA. The LLA prefix length is determined by adding 64 to the MNP prefix length. e.g., for the MNP 2001:db8:1000:2000::/56 the corresponding LLA-MNP prefix is fe80::2001:db8:1000:2000/120. (The base LLA-MNP for each "/N" prefix sets the final 128-N bits to 0, but all LLA-MNPs that match the prefix are also accepted.) Non-MNP IPv6 prefix-based LLAs are also represented the same as for LLA-MNPs, but include a GUA prefix that is not properly covered by the MSP.

* IPv4-Compatible LLA-MNPs are constructed as fe80::{IPv4-Prefix}, i.e., the IID consists of 32 '0' bits followed by a 32 bit IPv4 address/prefix, which may be either public or private in correspondence with the network layer addressing plan. The IPv4-Compatible LLA-MNP prefix length is determined by adding 96 to the IPv4 prefix length. For example, the IPv4-Compatible LLA-MNP for 192.0.2.0/24 is fe80::192.0.2.0/120, also written as fe80::c000:0200/120. (The base LLA-MNP for each "/N" prefix sets the final 128-N bits to 0, but all LLA-MNPs that match the prefix are also accepted.) Non-MNP IPv4 prefix-based LLAs are also represented the same as for LLA-MNPs, but include a GUA prefix that is not properly covered by the MSP.

* LLA-RNDs are randomly-generated and assigned to Proxy/Servers and other SRT infrastructure elements. They may also be assigned by Clients to support the MNP delegation process. The upper 72 bits of the LLA-RND encode the prefix fe80::/72, and the lower 56 bits include a randomly-generated candidate pseudo-random value configured as specified in [RFC7217]; if the most significant 24 bits of the 56 bit candidate encodes the value '0', the node generates a new candidate to obtain one with a different most significant 24 bits to avoid overlap with IPv4-Compatible LLAs.

* The address fe80::/128 (i.e., the LLA /64 prefix followed by an all-zero IID) is considered the LLA Subnet Router Anycast address

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 RND LLA constructs discussed above.

Since LLA-MNPs are based on the distribution of administratively assured unique MNPs, and since LLA-RNDs are assumed unique through pseudo-random 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 LLA-MNP. (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, this would interfere with the relationship between OMNI LLAs and ULAs (see: Section 9) and render many OMNI functions inoperable. The analysis provided in [RFC7421] furthermore suggests that the 64-bit boundary will remain in the IPv6 architecture for the foreseeable future.

9. Unique-Local Addresses (ULAs)

OMNI links use IPv6 Unique-Local Addresses (ULAs) as the source and destination addresses in both IPv6 ND messages and OAL packet IPv6 encapsulation headers. ULAs are routable only within the scope of an OMNI link, and are derived from the IPv6 Unique Local Address prefix fd00::/8 (i.e., the prefix fc00::/7 followed by the L bit set to 1). When the first 16 bits of the ULA encode the value fd00::/16, the address is considered as either a Temporary ULA (TLA) or an eXtended ULA (XLA) - see below. For all other ULAs, the 56 bits following fd00::/8 encode a 40-bit Global ID followed by a 16-bit Subnet ID as specified in Section 3 of [RFC4193]. All OMNI link ULA types finally include a 64-bit value in the IID portion of the address ULA::/64 as specified below.

When a node configures a ULA for OMNI, it selects a 40-bit Global ID for the OMNI link initialized to a candidate pseudo-random value as specified in Section 3 of [RFC4193]; if the most significant 8 bits of the candidate encodes the value '0', the node selects a new candidate until it obtains one with a different most significant 8 bits. All nodes on the same OMNI link use the same Global ID, and statistical uniqueness of the pseudo-random Global ID provides a unique OMNI link identifier allowing different links to be joined together in the future without requiring renumbering.

Next, for each logical segment of the same OMNI link the node selects a 16-bit Subnet ID value between 0x0000 and 0xffff. Nodes on the same logical segment configure the same Subnet ID, but nodes on different segments of the same OMNI link can still exchange IPv6 ND messages as single-hop neighbors even if they configure different Subnet IDs. When a node moves to a different OMNI link segment, it resets the Global ID and Subnet ID value according to the new segment but need not change the IID.

ULAs and their associated prefix lengths are configured in correspondence with LLAs through stateless prefix translation where "ULA-MNPs" simply copy the IIDs of their corresponding LLA-MNPs and "ULA-RNDs" simply copy the IIDs of their corresponding LLA-RNDs. For example, for the OMNI link ULA prefix fd{Global}:{Subnet}::/64:

  *the ULA-MNP corresponding to the LLA-MNP fe80::2001:db8:1:2 with a 56-bit MNP length is simply fd{Global}:{Subnet}:2001:db8:1:2/120 (where, the ULA prefix length becomes 64 plus the IPv6 MNP length).

  *the ULA-MNP corresponding to fe80::192.0.2.0 with a 28-bit MNP length is simply fd{Global}:{Subnet}::192.0.2.0/124 (where, the ULA prefix length becomes 96 plus the IPv4 MNP length).

  *the ULA-RND corresponding to fe80::0012:3456:789a:bcde is simply fd{Global}:{Subnet}::0012:3456:789a:bcde/128.

  *the Subnet Router Anycast ULA corresponding to fe80::/128 is simply fd{Global}:{Subnet}::/128.

The ULA presents an IPv6 address format that is routable within the OMNI link routing system and can be used to convey link-scoped (i.e., single-hop) IPv6 ND messages across multiple hops through IPv6 encapsulation [RFC2473]. The OMNI link extends across one or more underlying Internetworks to include all Proxy/Servers and other service nodes. All Clients are also considered to be connected to the OMNI link, however unnecessary encapsulations are omitted whenever possible to conserve bandwidth (see: Section 14).

Clients can configure TLAs when they have no other ULA addresses by setting the ULA prefix to fd00::/16 followed by a 48-bit randomly-generated number followed by a random or MNP-based IID as discussed in Section 8. XLAs are a special-case TLA that use the prefix fd00::/64. (Note that XLAs can also be formed from LLAs simply by inverting bits 7 and 8 of 'fe80' to form 'fd00'.)

OMNI nodes use XLA-MNPs as "default" ULAs for representing MNPs in the OMNI link routing system. Clients use {TLA,XLA}-MNPs when they already know their MNP but need to express it outside the context of a specific ULA prefix, and Proxy/Servers advertise XLA-MNPs into the OMNI link routing system instead of advertising fully-qualified {TLA,ULA}-MNPs and/or non-routable LLA-MNPs.

{TLAs,XLAs} provide initial "bootstrapping" addresses while the Client is in the process of procuring an MNP and/or identifying the ULA prefix for the OMNI link segment; TLAs are not advertised into the OMNI link routing system but can be used for Client-to-Client communications within a single {A,I,E}NET when no OMNI link infrastructure is present. Within each individual {A,I,E}NET, TLAs employ optimistic DAD principles [RFC4429] since they are statistically unique.

Each OMNI link may be subdivided into SRT segments that often correspond to different administrative domains or physical partitions. Each SRT segment is identified by a different Subnet ID within the same ULA ::/48 prefix. Multiple distinct OMNI links with different ULA ::/48 prefixes can also be joined together into a single unified OMNI link through simple interconnection without requiring renumbering. In that case, the (larger) unified OMNI link routing system may carry multiple distinct ULA prefixes.

OMNI nodes can use Segment Routing [RFC8402] 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-6man-aero].

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/MNP addressing under certain limiting conditions (see: Section 10). When used within the context of OMNI, ULAs based on the prefix fc00::/8 are referred to as "ULA-C's".

Note: When they appear in the OMNI link routing table, ULA-RNDs always use prefix lengths between /48 and /64 (or, /128) while XLA-MNPs always use prefix lengths between /65 and /128. {TLA,ULA}-MNPs and {TLA,XLA}-RNDs should never appear in the OMNI link routing table, but may appear in {A,I,E}NET routing tables.

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 Clients. Fixed correspondent node networks reachable from the OMNI link 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 MSPs are assigned by IANA [IPV6-GUA] and/or an associated Regional Internet Registry (RIR) such that the OMNI link can be interconnected to the global IPv6 Internet without causing inconsistencies in the routing system. An OMNI link could instead use ULAs with the 'L' bit set to 0 (i.e., from the "ULA-C" prefix fc00::/8)[RFC4193], however this would require IPv6 NAT if the domain were ever connected to the global IPv6 Internet.

For IPv4, GUA MSPs are assigned by IANA [IPV4-GUA] and/or an associated RIR such that the OMNI link can be interconnected to the global IPv4 Internet without causing routing inconsistencies. An OMNI ANET/ENET could instead use private IPv4 prefixes (e.g., 10.0.0.0/8, etc.) [RFC3330], however this would require IPv4 NAT at the INET-to-ANET/ENET boundary. OMNI interfaces advertise IPv4 MSPs into IPv6 routing systems as IPv4-Compatible IPv6 prefixes [RFC4291] (e.g., the IPv6 prefix for the IPv4 MSP 192.0.2.0/24 is :: 192.0.2.0/120).

OMNI interfaces assign the IPv4 anycast address TBD3 (see: IANA Considerations), and IPv4 routers that configure OMNI interfaces advertise the prefix TBD3/N into the routing system of other networks (see: IANA Considerations). OMNI interfaces also configure global IPv6 anycast addresses formed according to [RFC3056] as:

2002:TBD3{32}:MSP{64}:Link-ID{16}

where TBD3{32} is the 32 bit IPv4 anycast address and MSP{64} encodes an MSP zero-padded to 64 bits (if necessary). For example, the OMNI IPv6 anycast address for MSP 2001:db8::/32 is 2002:TBD3{32}:2001:db8:0:0:{Link-ID}, the OMNI IPv6 anycast address for MSP 192.0.2.0/24 is 2002:TBD3{32}::c000:0200:{Link-ID}, etc.).

The16-bit Link-ID in the OMNI IPv6 anycast address identifies a specific OMNI link within the domain that services the MSP. The special Link-ID value '0' is a wildcard that matches all links, while all other values identify specific links. Mappings between Link-ID values and the ULA Global IDs assigned to OMNI links are outside the scope of this document.

OMNI interfaces assign OMNI IPv6 anycast addresses, and IPv6 routers that configure OMNI interfaces advertise the corresponding prefixes

into the routing systems of other networks. An OMNI IPv6 anycast prefix is formed the same as for any IPv6 prefix; for example, the prefix 2002:TBD3{32}:2001:db8::/80 matches all OMNI IPv6 anycast addresses covered by the prefix. When IPv6 routers advertise OMNI IPv6 anycast prefixes in this way, Clients can locate and associate with either a specific OMNI link or any OMNI link within the domain that services the MSP of interest.

OMNI interfaces use OMNI IPv6 and IPv4 anycast addresses to support Service Discovery in the spirit of [RFC7094], i.e., the addresses are not intended for use in long-term transport protocol sessions. Specific applications for OMNI IPv6 and IPv4 anycast addresses are discussed throughout the document as well as in [I-D.templin-6man-aero].

11. Node Identification

OMNI Clients and Proxy/Servers that connect over open Internetworks include a unique node identification value for themselves in the OMNI options of their IPv6 ND messages (see: Section 12.2.12). An example identification value alternative is the Host Identity Tag (HIT) as specified in [RFC7401], while Hierarchical HITs (HHITs) [I-D.ietf-drip-rid] may be more appropriate for certain domains such as the Unmanned (Air) Traffic Management (UTM) service for Unmanned Air Systems (UAS). Another example 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 Client is truly outside the context of any infrastructure, it may have no MNP information at all. In that case, the Client can use a TLA 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 Client 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 Client connects via a protected-spectrum ANET, an alternate form of node identification (e.g., MAC address, serial number, airframe identification value, VIN, etc.) embedded in a ULA may be sufficient. The Client can then include OMNI "Node Identification" sub-options (see: Section 12.2.12) in IPv6 ND messages should the need to transmit identification information over the network arise.

12. Address Mapping - Unicast

OMNI interfaces maintain a network layer conceptual neighbor cache per [RFC1256] or [RFC4861] the same as for any IP interface, and (for IPv6) use the link-local address format specified in Section 8.

This network layer neighbor cache maintains state through static and/or dynamic configurations.

Each OMNI interface also maintains a separate internal OAL (adaptation layer) conceptual neighbor cache that includes a Neighbor Cache Entry (NCE) for each of its active OAL neighbors per [RFC4861]. Throughout this document, the terms "neighbor cache" and "NCE" refer to this OAL neighbor cache unless otherwise specified.

IPv6 Neighbor Discovery (ND) [RFC4861] messages sent over OMNI interfaces without OAL encapsulation observe the native underlay interface Source/Target Link-Layer Address Option (S/TLLAO) format (e.g., for Ethernet the S/TLLAO is specified in [RFC2464]). IPv6 ND messages sent from within the OMNI interface using OAL encapsulation do not include S/TLLAOs, but instead include a new option type that encodes interface attributes, traffic selectors and other OMNI link information. Hence, this document does not define a new S/TLLAO format but instead defines a new option type termed the "OMNI option" designed for these purposes. (Note that OMNI interface IPv6 ND messages sent without encapsulation may include both OMNI options and S/TLLAOs, but the information conveyed in each is mutually exclusive.)

OMNI interfaces prepare IPv6 ND messages that include one or more OMNI options (and any other IPv6 ND options) then completely populate all option information. If the OMNI interface includes an authentication signature, it sets the IPv6 ND message Checksum field to 0 and calculates the authentication signature over the length of the entire OAL packet or super-packet (beginning with a pseudo-header of the IPv6 ND message IPv6 header) but does not calculate/include the IPv6 ND message checksum itself. Otherwise, the OMNI interface calculates the standard IPv6 ND message checksum over the entire OAL packet or super-packet and writes the value in the Checksum field. OMNI interfaces verify authentication and/or integrity of each IPv6 ND message received according to the specific check(s) included, and process the message further only following verification.

OMNI interface Clients such as aircraft typically have multiple 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 underlay interfaces in a single IPv6 ND message exchange. OMNI interfaces manage their dynamically-changing multilink profiles by including OMNI options in IPv6 ND messages as discussed in the following subsections.

12.1. The OMNI Option

OMNI options appear in IPv6 ND messages formatted as shown in Figure 12:

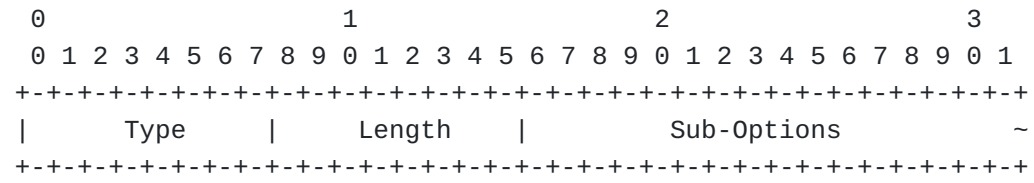


Figure 12: OMNI Option Format

In this format:

- Type is set to TBD4 (see: IANA Considerations).

- 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. If multiple OMNI option instances appear in the same IPv6 ND message, the union of the contents of all OMNI options is accepted unless otherwise qualified for specific sub-options below.

- Sub-Options is a Variable-length field padded if necessary such that the complete OMNI Option is an integer multiple of 8 octets long. Sub-Options contains zero or more sub-options as specified in Section 12.2.

The OMNI option is included in all OMNI interface IPv6 ND messages; the option is processed by receiving interfaces that recognize it and otherwise ignored. The OMNI interface processes all OMNI option instances received in the same IPv6 ND message in the consecutive order in which they appear. The OMNI option(s) included in each IPv6 ND message may include full or partial information for the neighbor. The OMNI interface therefore retains the union of the information in the most recently received OMNI options in the corresponding NCE.

12.2. OMNI Sub-Options

Each OMNI option includes a Sub-Options block containing zero or more individual sub-options. Each consecutive sub-option is concatenated immediately following its predecessor. All sub-options except Pad1 (see below) are in an OMNI-specific type-length-value (TLV) format encoded as follows:

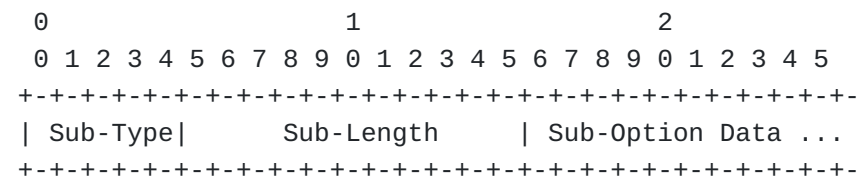


Figure 13: Sub-Option Format

*Sub-Type is a 5-bit field that encodes the sub-option type. Sub-option types defined in this document are:

Sub-Option Name	Sub-Type
Pad1	0
PadN	1
Neighbor Coordination	2
Interface Attributes	3
AERO Forwarding Parameters	4
Traffic Selector	5
Geo Coordinates	6
DHCPv6 Message	7
HIP Message	8
PIM-SM Message	9
Fragmentation Report	10
Node Identification	11
ICMPv6 Error	12
QUIC-TLS Message	13
Proxy/Server Departure	14
Sub-Type Extension	30

Figure 14

Sub-Types 15-29 are available for future assignment for major protocol functions, while Sub-Type 30 supports scalable extension to include other functions. Sub-Type 31 is reserved by IANA.

*Sub-Length is an 11-bit field that encodes the length of the Sub-Option Data in octets.

*Sub-Option Data is a block of data with format determined by Sub-Type and length determined by Sub-Length. Note that each individual sub-option may end on an arbitrary octet boundary, whereas the OMNI option itself must include padding if necessary for 8-octet alignment.

The OMNI interface codes each sub-option with a 2 octet header that includes Sub-Type in the most significant 5 bits followed by Sub-Length in the next most significant 11 bits. Each sub-option encodes a maximum Sub-Length value of 2038 octets minus the lengths of the OMNI option header and any preceding sub-options. This allows ample Sub-Option Data 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.

The OMNI interface codes initial sub-options in a first OMNI option instance and subsequent sub-options in additional instances in the same IPv6 ND message in the intended order of processing. The OMNI interface can then code any remaining sub-options in additional IPv6 ND messages if necessary. Implementations must observe these size limits and refrain from sending IPv6 ND messages larger than the OMNI interface MTU.

The OMNI interface processes all OMNI option Sub-Options received in an IPv6 ND message 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 an individual sub-option length would cause processing to exceed the OMNI option instance and/or IPv6 ND message lengths, the OMNI interface accepts any sub-options already processed and ignores the remainder of that instance. The interface then processes any remaining OMNI option instances in the same fashion to the end of the IPv6 ND message.

When an OMNI interface includes an authentication sub-option (e.g., see: Section 12.2.9), it MUST appear as the first sub-option of the first OMNI option which must appear immediately following the IPv6 ND message header (all other authentication sub-options are ignored). If the IPv6 ND message is the first packet in a combined OAL super-packet, the OMNI interface calculates the authentication signature over the entire length of the super-packet, i.e., and not just to the end of the IPv6 ND message itself. When the first sub-option is not authentication, the OMNI interface instead calculates the IPv6 ND message checksum over the entire length of the packet/super-packet.

When a Client OMNI interface prepares a secured unicast RS message, it includes a single Interface Attributes sub-option specific to the underlay interface that will transmit the RS (see: Section 12.2.4) immediately following the authentication and header extension sub-options if present; otherwise as the first sub-option of the first OMNI option which must appear immediately following the IPv6 ND message header. When a Client OMNI interface prepares a secured unicast NS message, it can instead include an AERO Forwarding Parameters sub-option specific to the underlay interface that will transmit the NS (see: Section 12.2.5).

Note: large objects that exceed the maximum Sub-Option Data length are not supported under the current specification; if this proves to be limiting in practice, future specifications may define support for fragmenting large sub-options across multiple OMNI options within the same IPv6 ND message (or even across multiple IPv6 ND messages, if necessary).

The following sub-option types and formats are defined in this document:

12.2.1. Pad1

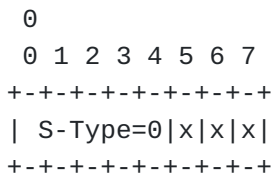


Figure 15: Pad1

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

*Sub-Type is followed by 3 'x' bits, set to any value on transmission (typically all-zeros) and ignored on reception. 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.

If more than one octet of padding is required, the PadN option, described next, should be used, rather than multiple Pad1 options.

12.2.2. PadN

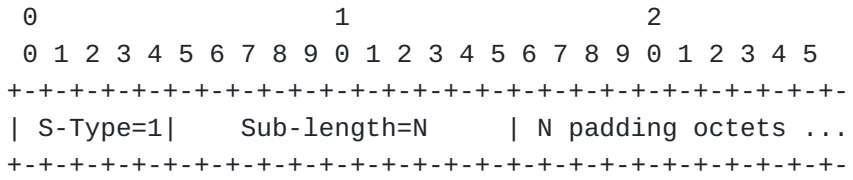


Figure 16: PadN

*Sub-Type is set to 1. If multiple instances appear in OMNI options of the same message all are processed.

*Sub-Length is set to N that encodes the number of padding octets that follow.

*Sub-Option Data consists of N octets, set to any value on transmission (typically all-zeros) and ignored on receipt.

When a proxy forwards an IPv6 ND message with OMNI options, it can employ PadN to void any sub-options (other than Pad1) that should not be processed by the next hop by simply writing the value '1' over the Sub-Type. When the proxy alters the IPv6 ND message contents in this way, any included authentication and integrity checks are invalidated. See: Appendix B for a discussion of IPv6 ND message authentication and integrity.

12.2.3. Neighbor Coordination

IPv6 ND messages used for Prefix Length assertion, service coordination and/or Window Synchronization include a Neighbor Coordination sub-option. If a Neighbor Coordination sub-option is included, it must appear immediately after the authentication sub-option if present; otherwise, as the first (non-padding) sub-option of the first OMNI option. If multiple Neighbor Coordination sub-options are included (whether in a single OMNI option or multiple), only the first is processed and all others are ignored.

The Neighbor Coordination sub-option is formatted as follows:

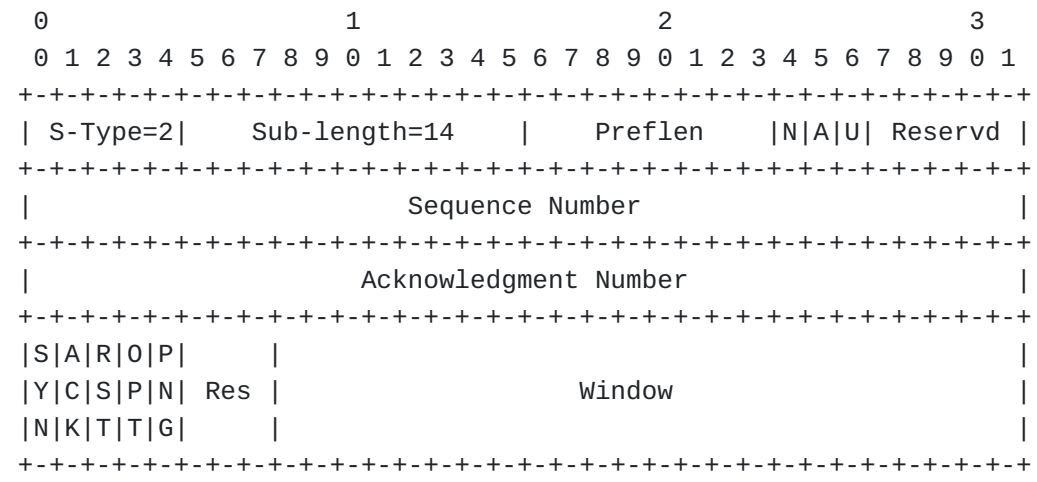


Figure 17: Neighbor Coordination

*Sub-Type is set to 2.

*Sub-Length is set to 14.

*The first two octets of Sub-Option Data contains a 1-octet Prefix Length followed by a 1-octet flags field interpreted as follows:

-Preflen is an 8 bit field that determines the length of prefix associated with a ULA containing an MNP. Values 0 through 128

specify a valid prefix length (if any other value appears the OMNI option must be ignored). For IPv6 ND messages sent from a Client to the MS, Preflen applies to the IPv6 source ULA and provides the length that the Client is requesting from or asserting to the MS. For IPv6 ND messages sent from the MS to the Client, Preflen applies to the IPv6 destination ULA and indicates the length that the MS is granting to the Client. For IPv6 ND messages sent between MS endpoints, Preflen provides the length associated with the source/target Client MNP that is subject of the ND message and encodes the value 64 plus the length of the MNP. (For example, if the MNP length is 56 then Preflen encodes the value 120.) When an IPv6 ND RS/RA message sets Preflen to 0, the recipient regards the message as a prefix release indication.

-The N/A/U flags are set or cleared in Client RS messages as directives to FHS and Hub Proxy/Servers and ignored in all other IPv6 ND messages. When an FHS Proxy/Server forwards or processes an RS with the N flag set, it responds directly to NS Neighbor Unreachability Detection (NUD) messages by returning NA(NUD) replies; otherwise, it forwards NS(NUD) messages to the Client. When the Hub Proxy/Server receives an RS with the A flag set, it responds directly to NS Address Resolution (AR) messages by returning NA(AR) replies; otherwise, it forwards NS(AR) messages to the Client. When the Hub Proxy/Server receives an RS with the U flag set, it maintains a Report List of recent NS(AR) message sources for the source or target Client and sends uNA messages to all list members if any aspects of the Client's underlay interfaces change. Proxy/Servers function according to the N/A/U flag settings received in the most recent RS message to support dynamic Client updates. In all IPv6 ND messages, the remaining 5 flag bits are set to 0 on transmission and ignored on reception.

*The remainder of Sub-Option Data contains a 4-octet Sequence Number, followed by a 4-octet Acknowledgement Number, followed by a 1-octet flags field followed by a 3-octet Window size modeled from the Transmission Control Protocol (TCP) header specified in Section 3.1 of [RFC0793]. The (SYN, ACK, RST) flags are used for TCP-like window synchronization, while the TCP (URG, PSH, FIN) flags are not used and therefore omitted. The (OPT, PNG) flags are OMNI-specific, and the remaining flags are Reserved. Together, these fields support the asymmetric and symmetric OAL window synchronization services specified in Section 6.6.

12.2.4. Interface Attributes

The Interface Attributes sub-option provides neighbors with forwarding information for the multilink conceptual sending algorithm discussed in Section 14. Neighbors use the forwarding information to selecting among potentially multiple candidate underlay interfaces that can be used to forward carrier packets to the neighbor based on factors such as traffic selectors and link quality. Interface Attributes further include link-layer address information to be used for either direct INET encapsulation for targets in the local SRT segment or spanning tree forwarding for targets in remote SRT segments.

OMNI nodes include Interface Attributes for some/all of a source or target Client's underlay interfaces in NS/NA and uNA messages used to publish Client information (see: [I-D.templin-6man-aero]). At most one Interface Attributes sub-option for each distinct ifIndex may be included; if an IPv6 ND message includes multiple Interface Attributes sub-options for the same ifIndex, the first is processed and all others are ignored. OMNI nodes that receive NS/NA messages can use all of the included Interface Attributes and/or Traffic Selectors to formulate a map of the prospective source or target node as well as to seed the information to be later populated in an AERO Forwarding Parameters sub-option (see: Section 12.2.5).

OMNI Clients and Proxy/Servers also include Interface Attributes sub-options in RS/RA messages used to initialize, discover and populate routing and addressing information. Each RS message MUST contain exactly one Interface Attributes sub-option with an ifIndex corresponding to the Client's underlay interface used to transmit the message, and each RA message MUST echo the same Interface Attributes sub-option with any (proxyed) information populated by the FHS Proxy/Server to provide operational context.

OMNI Client RS and Proxy/Server RA messages MUST include the Interface Attributes sub-option for the Client underlay interface in the first OMNI option immediately following the Neighbor Coordination and/or authentication sub-option(s) if present; otherwise, immediately following the OMNI header. When an FHS Proxy/Server receives an RS message destined to an anycast L2 address, it MUST include an Interface Attributes sub-option with ifIndex '0' that encodes its unicast L2 address relative to the Client's underlay interface immediately after the Client Interface Attributes sub-option in the solicited RA response. Any additional Interface Attributes sub-options that appear in RS/RA messages are ignored.

The Interface Attributes sub-option is formatted as shown below:

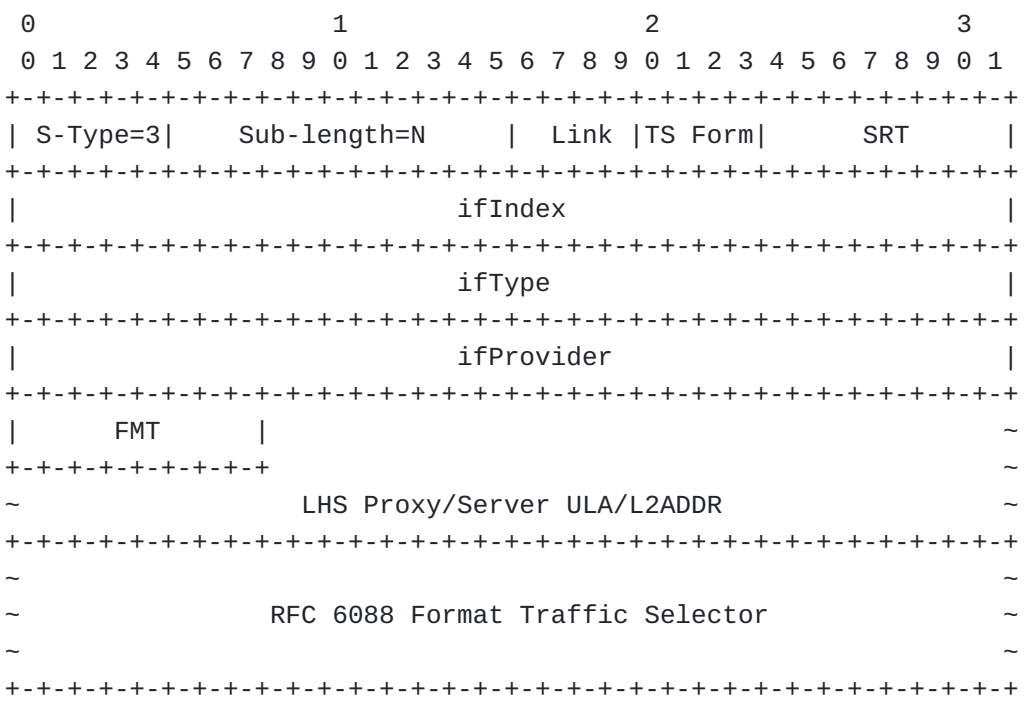


Figure 18: Interface Attributes

*Sub-Type is set to 3.

*Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow.

*Sub-Option Data contains an "Interface Attributes" option encoded as follows:

-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").

-TS-Form is a 4-bit field that encodes the same value that would appear in an [RFC6088] TS Format and determines the trailing RFC 6088 Format Traffic Selector type, if present. The following values are defined:

o0 - no traffic selector

o1 - IPv4 binary traffic selector

o2 - IPv6 binary traffic selector

o0 - 15 - reserved for future use

-SRT is a 1-octet Segment Routing Topology prefix length value between 0 and 128 that determines the prefix length associated with the LHS ULA.

-ifIndex is a 4-octet index value corresponding to a specific underlay interface. Client OMNI interfaces MUST number each distinct underlay interface with a non-zero ifIndex value assigned by network management per [RFC2863] and include the value in this field. The ifIndex value '0' denotes "unspecified".

-ifType is a 4-octet type value corresponding to this underlay interface. The value is coded per the 'IANAifType-MIB' registry [http://www.iana.org].

-ifProvider is a 4-octet provider identifier corresponding to this underlay interface. This document defines the single provider identifier value '0' (undefined). Future documents may define other values.

-FMT - a 1-octet "Forward/Mode/Type" code interpreted as follows:

  oThe most significant two bits (i.e., "FMT-Forward" and "FMT-Mode") are interpreted in conjunction with one another. When FMT-Forward is clear, the LHS Proxy/Server performs OAL reassembly and decapsulation to obtain the original IP packet before forwarding. If the FMT-Mode bit is clear, the LHS Proxy/Server then forwards the original IP packet at layer 3; otherwise, it invokes the OAL to re-encapsulate, re-fragment and forwards the resulting carrier packets to the Client via the selected underlay interface. When FMT-Forward is set, the LHS Proxy/Server forwards unsecured OAL fragments to the Client without reassembling, while reassembling secured OAL fragments before re-fragmenting and forwarding to the Client. If FMT-Mode is clear, all carrier packets destined to the Client must always be forwarded through the LHS Proxy/Server; otherwise the Client is eligible for direct forwarding over the open INET where it may be located behind one or more NATs.

  oThe value encoded in the least significant 6 bits (i.e., "FMT-Type") determines the type and length of the L2ADDR field as follows:

    o0 - L2ADDR is 4 octets in length and encodes an IPv4 address.

    o1 - L2ADDR is 16 octets in length and encodes an IPv6 address.

    o2 - L2ADDR is 6 octets in length and encodes an EUI-48 address [EUI].

o3 - L2ADDR is 8 octets in length and encodes an EUI-64 address [EUI].

o4-63 - Reserved for future use.

- LHS Proxy/Server ULA/L2ADDR - encodes the 15 least significant octets of the Proxy/Server ULA followed by the L2ADDR field formatted as above (note that the FMT code is replaced with the value "fd" after processing to form a proper 16 octet ULA). When SRT and ULA are both set to 0, the LHS Proxy/Server is considered unspecified in this IPv6 ND message. FMT, SRT and LHS together provide guidance for the OMNI interface forwarding algorithm. Specifically, if LHS::/SRT is located in the local OMNI link segment, then the source can address the target Client either through its dependent Proxy/Server or through direct encapsulation following NAT traversal according to FMT. Otherwise, the target Client is located on a different SRT segment and the path from the source must employ a combination of route optimization and spanning tree hop traversals. L2ADDR identifies the LHS Proxy/Server's INET-facing interface not located behind NATs, therefore no UDP port number is included since port number 8060 is used when the L2 encapsulation includes a UDP header. Instead, L2ADDR includes only an L2 address with type and length determined by FMT-Type as described above. When L2ADDR includes an IPv4 or IPv6 address, it is recorded in network byte order in ones-compliment "obfuscated" form as specified in [RFC4380].

- RFC 6088 Format Traffic Selector - traffic selectors formatted according to TS Form, with length determined by the remainder of the sup-option length following the LHS information. When TS Form encodes the value 1 or 2, the field is processed per [RFC6088]; when TS Form encodes any other value the field (if present) is ignored.

12.2.5. AERO Forwarding Parameters

OMNI nodes include the AERO Forwarding Parameters sub-option in NS/NA messages used to coordinate with multilink route optimization targets. If an NS/NA message includes the sub-option in a manner that solicits a response, the NA response must also include the sub-option. The OMNI node MUST include the sub-option in the first OMNI option immediately following the Neighbor Coordination and/or authentication message sub-option(s) if present. Otherwise, the OMNI node MUST include the sub-option immediately following the OMNI header. Each NS/NA message may contain at most one AERO Forwarding Parameters sub-option; if an NS/NA message contains additional AERO Forwarding Parameters sub-options, the first is processed and all others are ignored.

When an NS/NA message includes an AERO Forwarding Parameters sub-option with Job code '00' (see below), the FHS Client Interface Attributes MUST correspond to the underlay interface used to transmit the solicitation message. When the NS/NA message also includes Interface Attributes sub-options and/or Traffic Selectors, the options must appear following the AERO Forwarding Parameters sub-option.

The AERO Forwarding Parameters sub-option includes the necessary state for establishing AERO Forwarding Vectors (AFVs) in the AERO Forwarding Information Bases (AFIBs) of the OAL source, destination and intermediate nodes in the path. The sub-option also records addressing information for FHS/LHS nodes on the path, including "L2ADDRs" which MUST be unicast encapsulation addresses (i.e., and not anycast/multicast). The manner for populating multilink forwarding information is specified in detail in [I-D.templin-6man-aero].

The AERO Forwarding Parameters sub-option is formatted as shown in Figure 19:

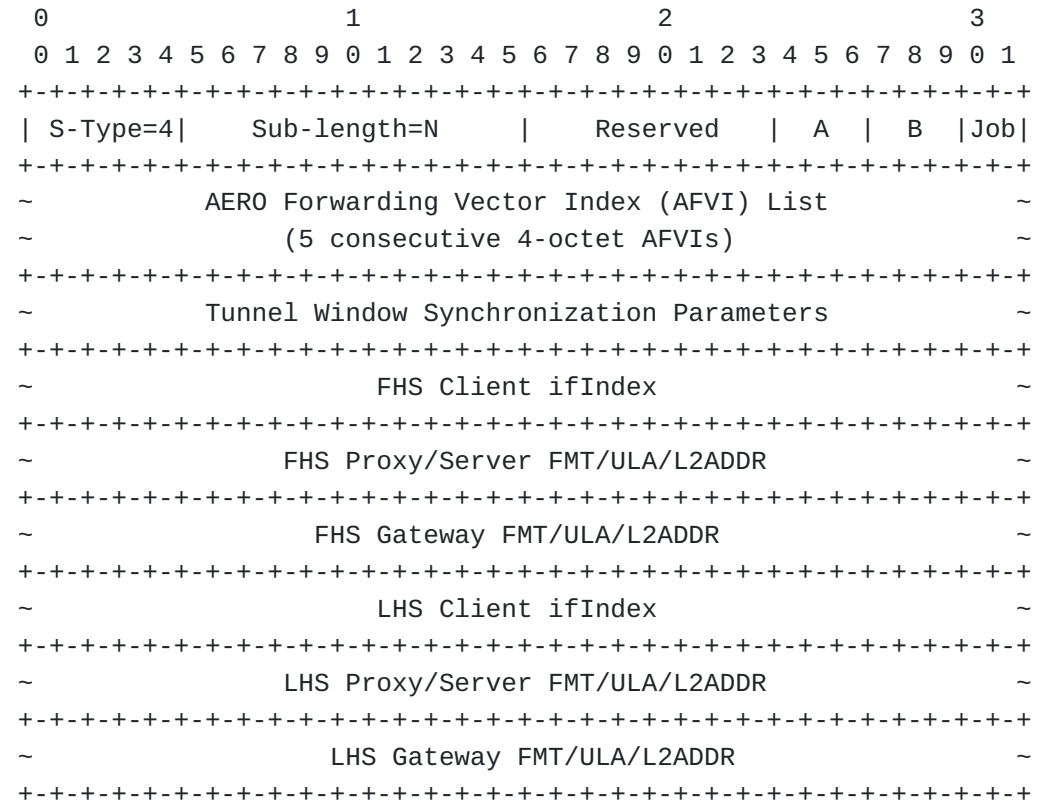


Figure 19: AERO Forwarding Parameters

*Sub-Type is set to 4. If multiple instances appear in the same message (i.e., whether in a single OMNI option or multiple) the first instance is processed and all others are ignored.

*Sub-Length encodes the number of Sub-Option Data octets that follow. The length includes all fields up to and including the Tunnel Window Synchronization Parameters for all Job codes, while including the remaining fields only for Job codes "0" and "1" (see below).

*Sub-Option Data contains AERO Forwarding Parameters as follows:

-Reserved is a 1-octet reserved field set to 0 on transmission and ignored on receipt.

-A/B and Job are fields that determine per-hop processing of the AFVI List, where A is a 3-bit count of the number of "A" AFVI List entries and B is a 3-bit count of the number of "B" AFVI List entries (valid A/B values are 0-5). Job is a 2-bit code interpreted as follows:

o'00' - "Initialize; Build B" - the FHS source sets this code in an NS/NA used to initialize AFV state (any other messages that include this code MUST be dropped). The FHS source first sets A/B to 0, and the FHS source and each intermediate node along the path to the LHS destination that processes the message creates a new AFV. Each node that processes the message then assigns a unique 4-octet "B" AFVI to the AFV and also writes the value into list entry B, then increments B. When the message arrives at the LHS destination, B will contain the number of AFVI List "B" entries, with the FHS source entry first, followed by entries for each consecutive intermediate node and ending with an entry for the final intermediate node (i.e., the list is populated in the forward direction). An NS/NA message containing a Job Code '00' AERO Forwarding Parameters sub-option always solicits a responsive NA message containing Job Code '01'.

o'01' - "Follow B; Build A" - the LHS source sets this code in a solicited NA response to an NS/NA with Job code "0" (any other messages that include this code MUST be dropped). The LHS source first copies the AFVI List and B value from the code '00' solicitation into these fields and sets A to 0. The LHS source and each intermediate node along the path to the FHS destination that processes the message then uses AFVI List entry B to locate the

corresponding AFV. Each node that processes the message then assigns a unique 4-octet "A" AFVI to the AFV and also writes the value into list entry B, then increments A and decrements B. When the message arrives at the FHS destination, A will contain the number of AFVI List "A" entries, with the LHS source entry last, preceded by entries for each consecutive intermediate node and beginning with an entry for the final intermediate node (i.e., the list is populated in the reverse direction).

o'10' - "Follow A; Record B" - the FHS node that sent the original code '00' solicitation and received the corresponding code '01' advertisement sets this code in any subsequent NS/NA messages sent to the same LHS destination. The FHS source copies the AFVI List and A value from the code '01' advertisement into these fields and sets B to 0. The FHS source and each intermediate node along the path to the LHS destination that processes the message then uses the "A" AFVI found at list entry B to locate the corresponding AFV. Each node that processes the message then writes the AFV's "B" AFVI into list entry B, then decrements A and increments B. When the message arrives at the LHS destination, B will contain the number of AFVI List "B" entries populated in the forward direction.

o'11' - "Follow B; Record A" - the LHS node that received the original code '00' solicitation and sent the corresponding code '01' advertisement sets this code in any subsequent NS/NA messages sent to the same FHS destination. The LHS source copies the AFVI List and B values from the code '00' solicitation into these fields and sets A to 0. The LHS source and each intermediate node along the path to the FHS destination that processes the message then uses the "B" AFVI List entry found at list entry B to locate the corresponding AFV. Each node that processes the message then writes the AFV's "A" AFVI into list entry B, then increments A and decrements B. When the message arrives at the FHS destination, A will contain the number of AFVI List "A" entries populated in the reverse direction.

Job and A/B together determine the per-hop behavior at each FHS/LHS source, intermediate node and destination that processes an IPv6 ND message. When a Job code specifies "Initialize", each FHS/LHS node that processes the message creates a new AFV. When a Job code specifies "Build", each node that processes the message assigns a new AFVI. When a Job code specifies "Follow", each node that processes the message uses an A/B AFVI List entry to locate an AFV (if the AFV cannot be located, the node returns a parameter problem and

drops the message). Using this algorithm, FHS sources that send code '00' solicitations and receive code '01' advertisements discover only "A" information, while LHS sources that receive code '00' solicitations and return code '01' advertisements discover only "B" information. FHS/LHS intermediate nodes can instead examine A, B and the AFVI List to determine the number of previous hops, the number of remaining hops, and the A/B AFVIs associated with the previous/remaining hops. However, no intermediate nodes will discover inappropriate A/B AFVIs for their location in the multihop forwarding chain. See: [I-D.templin-6man-aero] for further discussion on A/B AFVI processing.

- AERO Forwarding Vector Index (AFVI) List is a 20-octet block that contains 5 consecutive 4-octet AFVI entries. The FHS/LHS source and each intermediate node on the path to the destination processes the list according to the Job and A/B codes (see above). Note that the reason the AFVI list contains at most 5 entries is that only the FHS (Client, Proxy/Server, Gateway) and LHS (Client, Proxy/Server, Gateway) nodes are eligible for OMNI link route optimization resulting in at most 5 AFVIs "hops" that must be exposed. All other OMNI link nodes (i.e., downstream Clients that connect via an FHS/LHS Client) must forward through their upstream-dependent OMNI link neighbors without applying OMNI link route optimization.

- Tunnel Window Synchronization Parameters is a 12-octet block that consists of a 4-octet Sequence Number followed by a 4-octet Acknowledgement Number followed by a 1-octet Flags field followed by a 3-octet Window field (i.e., the same as for the OMNI header parameters). Tunnel endpoints use these parameters for simultaneous middlebox window synchronization in a single solicitation/advertisement message exchange.

- For Job codes '00' and '01' only, two trailing state variable blocks are included for First-Hop Segment (FHS) followed by Last-Hop Segment (LHS) network elements. When present, each block encodes the following information:

  - Client ifIndex encodes the 4 octet index for this Client interface. The source sets the FHS/LHS ifIndex values according to its own local interface information and neighbor information discovered from earlier NS/NA address resolution exchanges.

  - Proxy/Server FMT/ULA/L2ADDR encodes a 1 octet FMT code immediately followed by the 15 least significant octets of the Proxy/Server ULA, where FMT/ULA are interpreted the same as defined for the Interface Attribute sub-option in

Section 12.2.4 but with the FMT-Forward and FMT-Mode bits ignored. FMT/ULA is then followed by a 16 octet L2ADDR that identifies an open INET interface not located behind NATs, therefore no UDP port number is included since port number 8060 is used when the L2 encapsulation includes a UDP header. Unlike the Interface Attribute sub-option, L2ADDR is always exactly 16 octets in length regardless of the actual L2 address length 'N' with the L2 address appearing in the N least-significant octets and the (16 - N) most-significant octets set to '0'. When L2ADDR includes an IPv4 or IPv6 address, it is recorded in network byte order in ones-compliment "obfuscated" form as specified in [RFC4380].

oGateway FMT/ULA/L2ADDR encodes a 1 octet FMT code followed by the 15 least significant ULA octets followed by a 16 octet L2ADDR exactly as for the Proxy/Server FMT/ULA/L2ADDR above.

12.2.6. Traffic Selector

The Traffic Selector sub-option provides forwarding information for the multilink conceptual sending algorithm discussed in Section 14. The sub-option includes the same information that would appear in an Interface Attributes sub-option; hence, it can be used as an extension to any Interface Attributes with the same ifIndex value present.

IPv6 ND messages may include Traffic Selectors for some or all of the source/target Client's underlay interfaces (see: [I-D.templin-6man-aero] for more information). Multiple Traffic Selector sub-options with the same ifIndex value may appear in the same IPv6 ND message.

Traffic Selectors must be honored by all implementations in the format shown below:

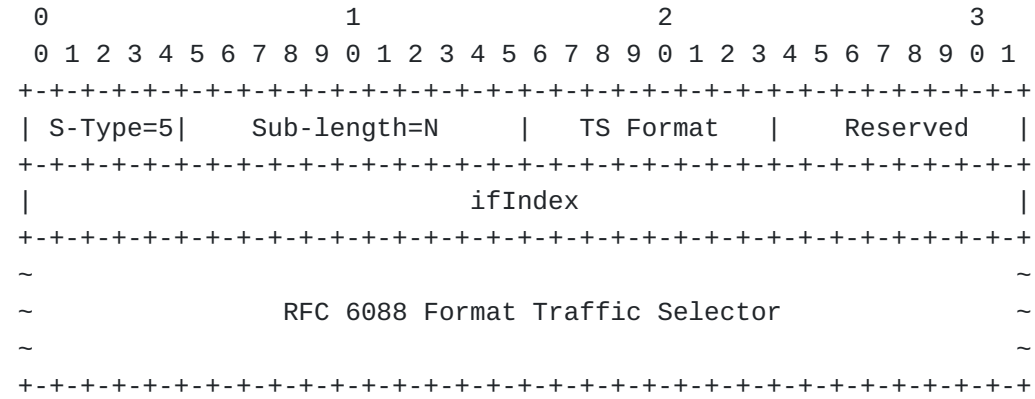


Figure 20: Traffic Selector

*Sub-Type is set to 5. Each IPv6 ND message may contain zero or more Traffic Selectors for each ifIndex; when multiple Traffic Selectors for the same ifIndex appear, all are processed and the cumulative information from all is accepted.

*Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow.

*Sub-Option Data contains a "Traffic Selector" encoded as follows:

-TS Format is a 1-octet field that encodes a Traffic Selector version per [RFC6088]. If TS Format encodes the value 1 or 2, the Traffic Selector includes IPv4 or IPv6 information, respectively. If TS Format encodes any other value, the sub-option is ignored.

-Reserved is a 1-octet field set to 0 on transmission and ignored on receipt

-ifIndex is a 4-octet value corresponding to a specific underlay interface the same as specified above for Interface Attributes and AERO Forwarding Parameters above. The OMNI options of a single message may include multiple Traffic Selector sub-options; each with the same or different ifIndex values.

-The remainder of the sub-option includes a traffic selector formatted per [RFC6088] beginning with the "Flags (A-N)" field, and with the Traffic Selector IP protocol version coded in the TS Format field. If a single interface identified by ifIndex requires Traffic Selectors for multiple IP protocol versions, or if a Traffic Selector block would exceed the available space, the remaining information is coded in additional Traffic Selector sub-options that all encode the same ifIndex.

12.2.7. Geo Coordinates

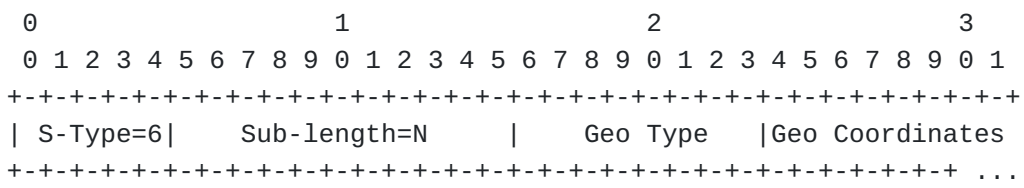


Figure 21: Geo Coordinates Sub-option

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

*Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow.

*Geo Type is a 1 octet field that encodes a type designator that determines the format and contents of the Geo Coordinates field that follows. The following types are currently defined:

  -0 - NULL, i.e., the Geo Coordinates field is zero-length.

*A set of Geo Coordinates of length up to the remaining available space for this OMNI option. New formats to be specified in future documents and may include attributes such as latitude/longitude, altitude, heading, speed, etc.

12.2.8. 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 Client RS messages and Proxy/Server RA messages. FHS Proxy/Servers that forward RS/RA messages between a Client and an LHS Proxy/Server also forward DHCPv6 Sub-Options unchanged. Note that DHCPv6 messages do not include a Checksum field since integrity is protected by the IPv6 ND message checksum, authentication signature and/or lower-layer authentication and integrity checks.

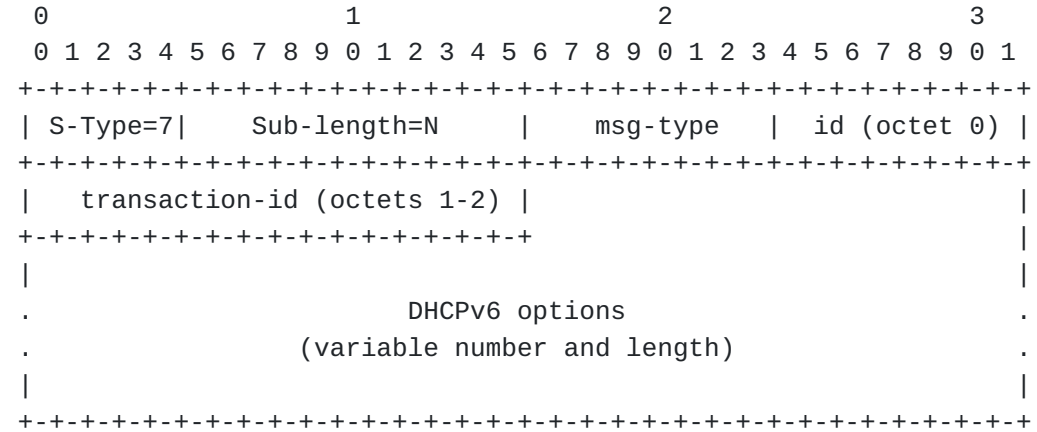


Figure 22: DHCPv6 Message Sub-option

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

*Sub-Length is set to N 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 limited by the remaining available space for this OMNI option.

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

*A set of DHCPv6 options coded according to Section 21 of [RFC8415] follows.

12.2.9. Host Identity Protocol (HIP) Message

The Host Identity Protocol (HIP) Message sub-option (when present) provides authentication for IPv6 ND messages exchanged between Clients and FHS Proxy/Servers over an open Internetwork. FHS Proxy/Servers authenticate the HIP authentication signatures in source Client IPv6 ND messages before securely forwarding them to other OMNI nodes. LHS Proxy/Servers that receive secured IPv6 ND messages from other OMNI nodes that do not already include a security sub-option insert HIP authentication signatures before forwarding them to the target Client.

OMNI interfaces MUST include the HIP message (when present) as the first sub-option of the first OMNI option, which MUST appear immediately following the IPv6 ND message header. OMNI interfaces can therefore easily locate the HIP message and verify the authentication signature without applying deep inspection. OMNI interfaces that receive IPv6 ND messages without a HIP (or other authentication) sub-option as the first OMNI sub-option instead verify the IPv6 ND message checksum.

OMNI interfaces include the HIP message sub-option when they forward IPv6 ND messages that require security over INET underlay interfaces, i.e., where authentication and integrity is not already assured by lower layers. The OMNI interface calculates the authentication signature over the entire length of the OAL packet (or super-packet) beginning with a pseudo-header of the IPv6 ND message header and extending over the remainder of the OAL packet. OMNI interfaces that process OAL packets that contain secured IPv6 ND messages verify the signature then either process the rest of the message locally or forward a proxyed copy to the next hop.

When a FHS Client inserts a HIP message sub-option in an NS/NA message destined to a target in a remote spanning tree segment, it must ensure that the insertion does not cause the message to exceed the OMNI interface MTU. When the remote segment LHS Proxy/Server forwards the NS/NA message from the spanning tree to the target Client, it inserts a new HIP message sub-option if necessary while

overwriting or cancelling the (now defunct) HIP message sub-option supplied by the FHS Client.

If the defunct HIP sub-option size was smaller than the space needed for the LHS Client HIP message (or, if no defunct HIP sub-option is present), the LHS Proxy/Server adjusts the space immediately following the OMNI header by copying the preceding portion of the IPv6 ND message into buffer headroom free space or copying the remainder of the IPv6 ND message into buffer tailroom free space. The LHS Proxy/Server then insets the new HIP sub-option immediately after the OMNI header and immediately before the next sub-option while properly overwriting the defunct sub-option if present.

If the defunct HIP sub-option size was larger than the space needed for the LHS Client HIP message, the LHS Proxy/Server instead overwrites the existing sub-option and writes a single Pad1 or PadN sub-option over the next 1-2 octets to cancel the remainder of the defunct sub-option. If the LHS Proxy/Server cannot create sufficient space through any means without causing the OMNI option to exceed 2040 octets or causing the IPv6 ND message to exceed the OMNI interface MTU, it returns a suitable error (see: Section 12.2.13) and drops the message.

The HIP message sub-option is formatted as shown below:

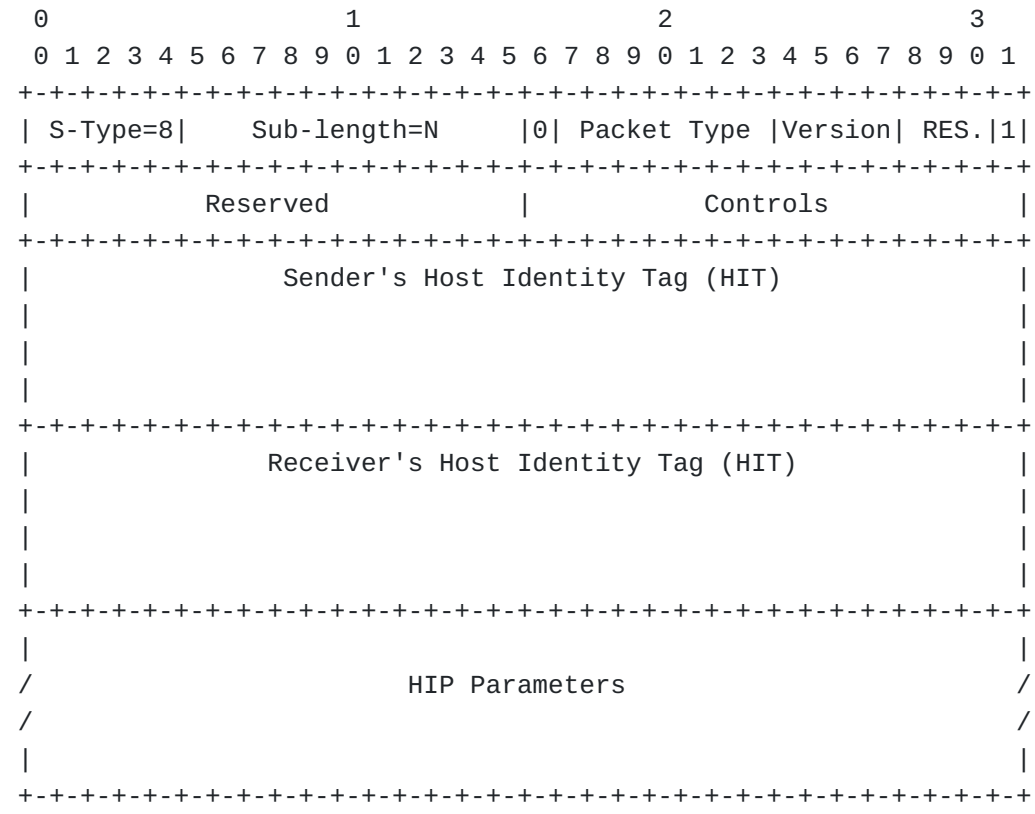


Figure 23: HIP Message Sub-option

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

*Sub-Length is set to N, 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 limited by the remaining available space for this OMNI option.

*The HIP message is coded per 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). Also, since the IPv6 ND message is already protected by the authentication signature and/or lower-layer authentication and integrity checks, the HIP message Checksum field is replaced by a Reserved field set to 0 on transmission and ignored on reception.

Note: In some environments, maintenance of a Host Identity Tag (HIT) namespace may be unnecessary for securely associating an OMNI node with an IPv6 address-based identity. In that case, IPv6 ULAs can be used instead of HITs in the authentication signature as long as the address can be uniquely associated with the Sender/Receiver.

12.2.10. PIM-SM Message

The Protocol Independent Multicast - Sparse Mode (PIM-SM) Message sub-option may be included in the OMNI options of IPv6 ND messages. PIM-SM messages are formatted as specified in Section 4.9 of [RFC7761], with the exception that the Checksum field is replaced by a Reserved field (set to 0) since the IPv6 ND message is already protected by the IPv6 ND message checksum, authentication signature and/or lower-layer authentication and integrity checks. The PIM-SM message sub-option format is shown in Figure 24:

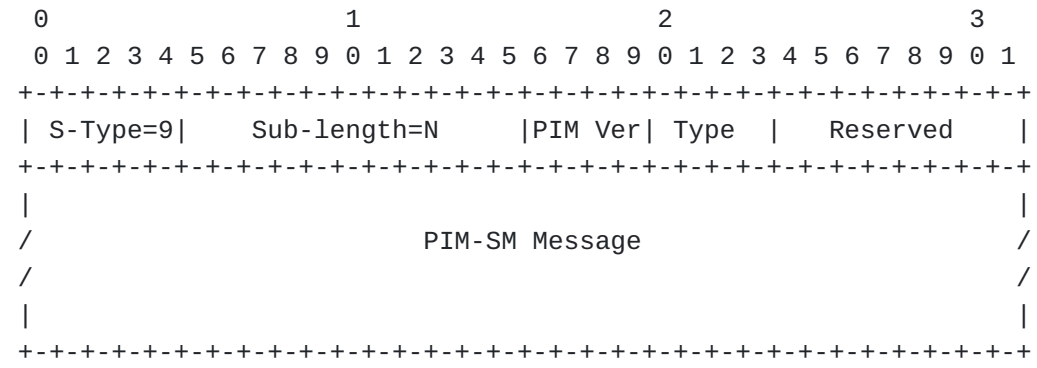


Figure 24: PIM-SM Message Option Format

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

*Sub-Length is set to N, i.e., the length of the option in octets beginning immediately following the Sub-Length field and extending to the end of the PIM-SM message. The length of the entire PIM-SM message is therefore limited by the remaining available space for this OMNI option.

*The PIM-SM message is coded exactly as specified in Section 4.9 of [RFC7761], except that the Checksum field is replaced by a Reserved field set to 0 on transmission and ignored on reception. The "PIM Ver" field MUST encode the value 2, and the "Type" field encodes the PIM message type. (See Section 4.9 of [RFC7761] for a list of PIM-SM message types and formats.)

12.2.11. Fragmentation Report (FRAGREP)

Fragmentation Report (FRAGREP) sub-options may be included in the OMNI options of uNA messages sent from an OAL destination to an OAL source. The message consists of (N / 20)-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.

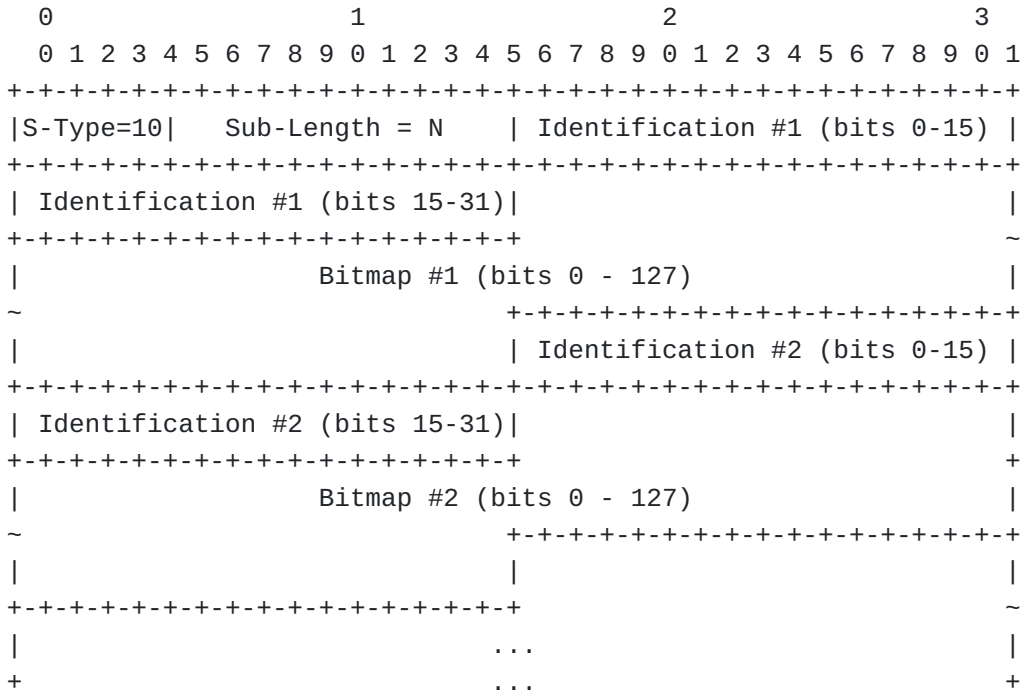


Figure 25: Fragmentation Report (FRAGREP)

*Sub-Type is set to 10. If multiple instances appear in OMNI options of the same message all are processed.

*Sub-Length is set to N, i.e., the length of the option in octets beginning immediately following the Sub-Length field and extending to the end of the sub-option. If N is not an integral multiple of 20 octets, the sub-option is ignored. The length of the entire sub-option should not cause the entire IPv6 ND message to exceed the minimum IPv6 MTU.

*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 may still be in transit.)

*Bitmap (i) includes an ordinal checklist of up to 128 fragments, with each bit set to 1 for a fragment received or 0 for a fragment missing. For example, for a 20-fragment OAL packet with ordinal fragments #3, #10, #13 and #17 missing and all other fragments received, Bitmap (i) encodes the following:

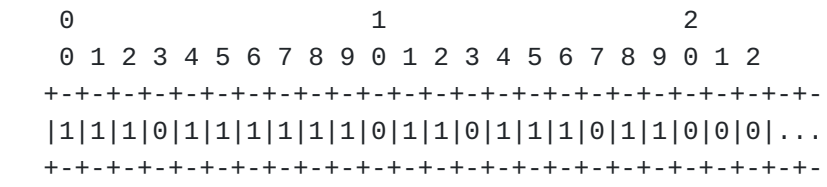


Figure 26

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

12.2.12. Node Identification

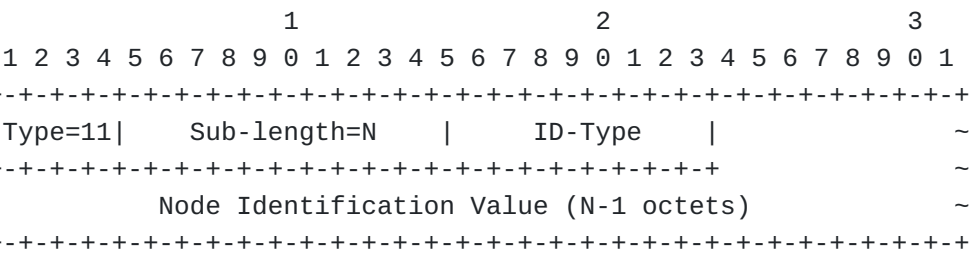


Figure 27: Node Identification

*Sub-Type is set to 11. 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. (It is therefore possible for a single IPv6 ND message to convey multiple distinct Node Identifications - each with a different ID-Type.)

*Sub-Length is set to N 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 limited by the remaining available space in this OMNI option.

*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 - IPv6 Address. Indicates that Node Identification contains a 16-octet IPv6 address that is not a (H)HIT. The IPv6 address type is determined according to the IPv6 addressing architecture [RFC4291].

-6 - 252 - Unassigned.

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

-255 - reserved by IANA.

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

OMNI interfaces code Node Identification Values used for DHCPv6 messaging purposes 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:

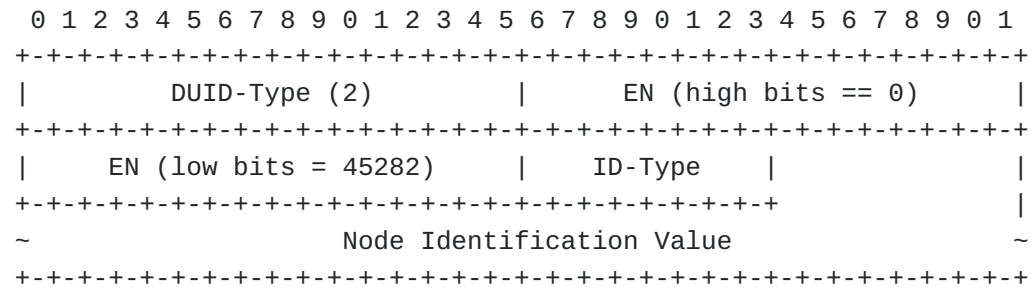


Figure 28: DUID-EN for OMNI Format

In this format, the OMNI interface codes the ID-Type and Node Identification Value fields from the OMNI sub-option following a 6 octet DUID-EN header, then includes the entire "DUID-EN for OMNI" in a DHCPv6 message per [RFC8415].

12.2.13. ICMPv6 Error

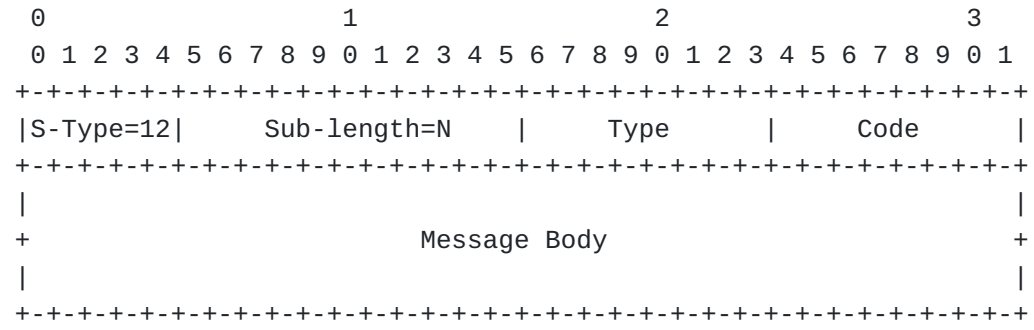


Figure 29: ICMPv6 Error

*Sub-Type is set to 12. If multiple instances appear in OMNI options of the same IPv6 ND message all are processed.

*Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow.

*Sub-Option Data includes a one octet Type followed by a one octet Code followed by an (N-2)-octet Message Body encoded exactly as per Section 2.1 of [RFC4443]. OMNI interfaces include as much of the ICMPv6 error message body in the sub-option as possible without causing the entire IPv6 ND message to exceed the minimum IPv6 MTU. While all ICMPv6 error message types are supported, OAL destinations in particular may include ICMPv6 PTB messages in uNA messages to provide MTU feedback information via the OAL source

(see: Section 6.8). Note: ICMPv6 informational messages must not be included and must be ignored if received.

12.2.14. QUIC-TLS Message

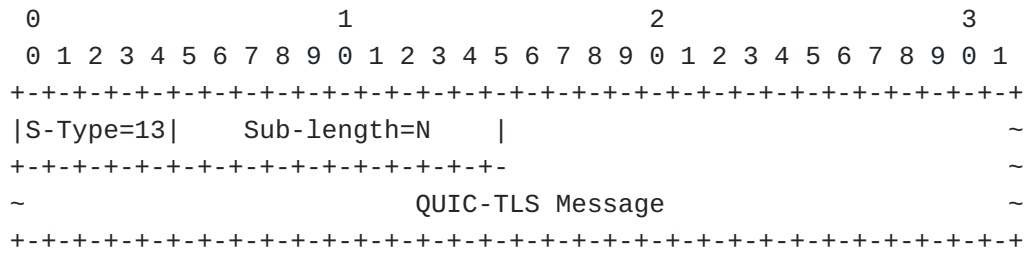


Figure 30: QUIC-TLS Message

*Sub-Type is set to 13. If multiple instances appear in OMNI options of the same IPv6 ND message, the first is processed and all others are ignored.

*Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow.

*The QUIC-TLS message [RFC9000][RFC9001][RFC9002] encodes the QUIC and TLS message parameters necessary to support QUIC connection establishment.

When present, the QUIC-TLS Message sub-option MUST appear immediately after the header of the first OMNI option in the IPv6 ND message; if the sub-option appears in any other location it MUST be ignored. IPv6 ND solicitation and advertisement messages serve as couriers to transport the QUIC and TLS parameters necessary to establish a secured QUIC connection.

12.2.15. Proxy/Server Departure

OMNI Clients include a Proxy/Server Departure sub-option in RS messages when they associate with a new FHS and/or Hub Proxy/Server and need to send a departure indication to an old FHS and/or Hub Proxy/Server. The Proxy/Server Departure sub-option is formatted as shown below:

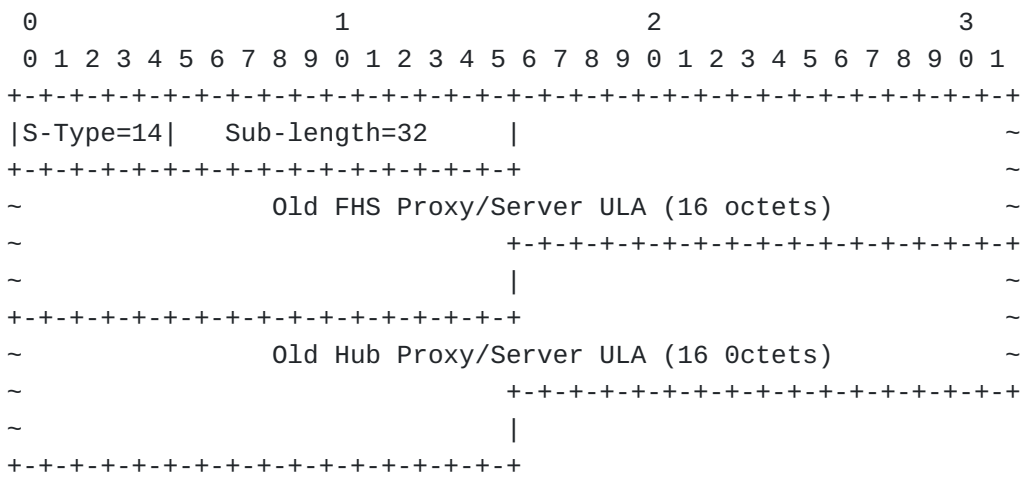


Figure 31: Proxy/Server Departure

*Sub-Type is set to 14.

*Sub-Length is set to 32.

*Sub-Option Data contains the 16 octet ULA for the "Old FHS Proxy/ Server" followed by a 16 octet ULA for an "Old Hub Proxy/Server. (If the Old FHS/Hub is unspecified, the corresponding ULA instead includes the value 0.)

12.2.16. 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 32:

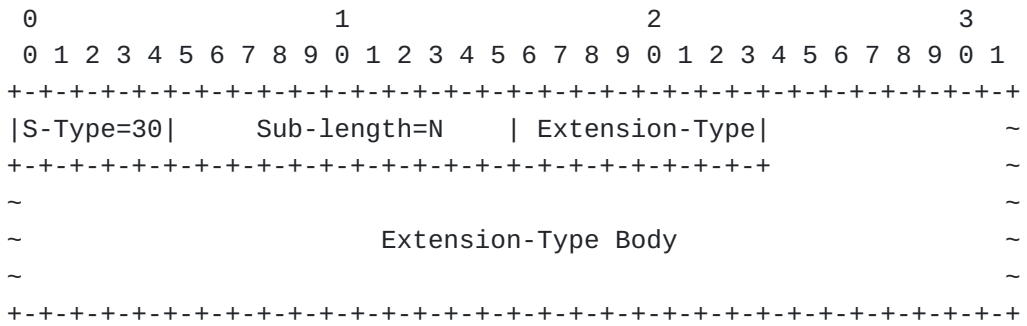


Figure 32: Sub-Type Extension

*Sub-Type is set to 30. If multiple instances appear in OMNI options of the same message all are processed, where each

individual extension defines its own policy for processing multiple of that type.

*Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow. The Extension-Type field is always present, and the maximum Extension-Type Body length is limited by the remaining available space in this OMNI option.

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

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

Extension-Type values 0 and 1 are defined in the following subsections, while 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.

12.2.16.1. RFC4380 Header Extension Option

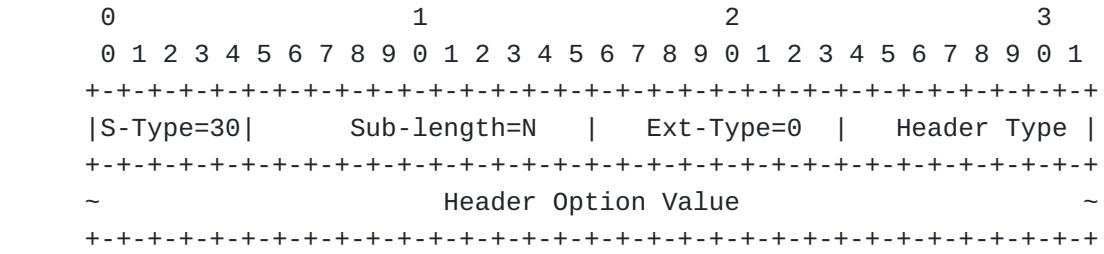


Figure 33: RFC4380 Header Extension Option (Extension-Type 0)

*Sub-Type is set to 30.

*Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow. The Extension-Type and Header Type fields are always present, and the Header Option Value is limited by the remaining available space in this OMNI option.

*Extension-Type is set to 0. Each instance encodes exactly one header option per Section 5.1.1 of [RFC4380], with Ext-Type and Header Type representing the first two octets of the option. 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. If Header Type indicates an Authentication Encapsulation (see below), the entire sub-option MUST appear as the first sub-option of the first OMNI option, which MUST appear immediately following the IPv6 ND message header.

*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 as a UDP port number followed by a 4-octet IPv4 address both in "obfuscated" form per Section 5.1.1 of [RFC4380].

  -1 - Authentication Encapsulation - value coded per Section 5.1.1 of [RFC4380].

  -2 - Origin Indication (IPv6) - value coded per Section 5.1.1 of [RFC4380], except that the address is a 16-octet IPv6 address instead of a 4-octet IPv4 address.

*Header Type values 3 through 252 are available for assignment by future specifications, which must also define the format of the Header Option Value and its processing rules. Header Type values 253 and 254 are reserved for experimentation, as recommended in [RFC3692], and value 255 is Reserved by IANA.

12.2.16.2. RFC6081 Trailer Extension Option

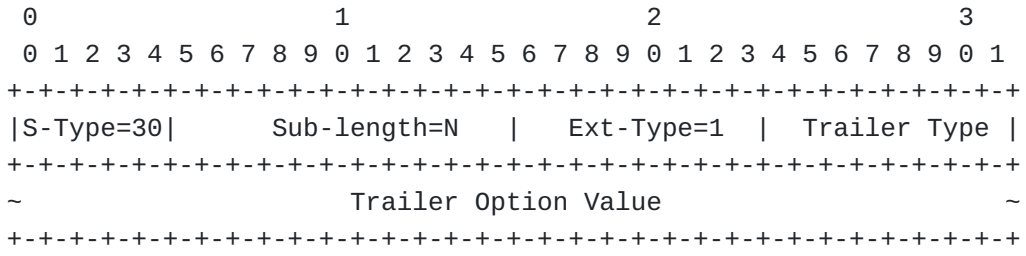


Figure 34: RFC6081 Trailer Extension Option (Extension-Type 1)

*Sub-Type is set to 30.

*Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow. The Extension-Type and Trailer Type fields are always present, and the maximum-length Trailer Option Value is limited by the remaining available space in this OMNI option.

*Extension-Type is set to 1. Each instance encodes exactly one trailer option per Section 4 of [RFC6081]. If multiple instances of the same Trailer Type appear in OMNI options of the same message the first instance is processed and all others ignored.

*Trailer Type and Trailer Option Value are coded exactly as specified in Section 4 of [RFC6081]; the following Trailer Types are currently defined:

-0 - Unassigned

-1 - Nonce Trailer - value coded per Section 4.2 of [RFC6081].

-2 - Unassigned

-3 - Alternate Address Trailer (IPv4) - value coded per Section 4.3 of [RFC6081].

-4 - Neighbor Discovery Option Trailer - value coded per Section 4.4 of [RFC6081].

-5 - Random Port Trailer - value coded per Section 4.5 of [RFC6081].

-6 - Alternate Address Trailer (IPv6) - value coded per Section 4.3 of [RFC6081], except that each address is a 16-octet IPv6 address instead of a 4-octet IPv4 address.

*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 underlay interface applies. The Client mobile router also serves as an IGMP/MLD Proxy for its ENETs and/or hosted applications per [RFC4605].

The Client uses Multicast Listener Discovery (MLDv2) [RFC3810] to coordinate with Proxy/Servers, and underlay network elements use MLD snooping [RFC4541]. The Client can also employ multicast routing protocols to coordinate with network-based multicast sources as specified in [I-D.templin-6man-aero].

Since the OMNI link model is NBMA, OMNI links support link-scoped multicast through iterative unicast transmissions to individual multicast group members (i.e., unicast/multicast emulation).

14. Multilink Conceptual Sending Algorithm

The Client's IPv6 layer selects the outbound OMNI interface according to SBM considerations when forwarding original IP packets from local or ENET applications to external correspondents. Each

OMNI interface maintains an internal OAL neighbor cache maintained the same as discussed in [RFC4861], but also includes additional state for multilink coordination. Each Client OMNI interface maintains default routes via Proxy/Servers discovered as discussed in Section 15, and may configure more-specific routes discovered through means outside the scope of this specification.

For each original IP packet it forwards, the OMNI interface selects one or more source underlay interfaces based on PBM factors (e.g., traffic attributes, cost, performance, message size, etc.) and one or more target underlay interfaces for the neighbor based on Interface Attributes received in IPv6 ND messages (see: Section 12.2.4). Multilink forwarding may also direct packet replication across multiple underlay interface pairs for increased reliability at the expense of duplication. The set of all Interface Attributes and Traffic Selectors received in IPv6 ND messages determines the multilink forwarding profile for selecting target underlay interfaces.

When the OMNI interface sends an original IP packet over a selected source underlay interface, it first employs OAL encapsulation and fragmentation as discussed in Section 5, then performs L2 encapsulation as directed by the appropriate AFV. The OMNI interface also performs L2 encapsulation (following OAL encapsulation) when the nearest Proxy/Server is located multiple hops away as discussed in Section 15.2.

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 underlay interfaces having diverse properties.

14.1. Multiple OMNI Interfaces

Clients may connect to multiple independent OMNI links within the same or different OMNI domains to support SBM. The Client configures a separate OMNI interface for each link so that multiple interfaces (e.g., omni0, omni1, omni2, etc.) are exposed to the IP layer. Each OMNI interface configures one or more OMNI anycast addresses (see: Section 10), and the Client injects the corresponding anycast prefixes into the ENET routing system. Multiple distinct OMNI links can therefore be used to support fault tolerance, load balancing, reliability, etc.

Applications in ENETs can use Segment Routing to select the desired OMNI interface based on SBM considerations. The application writes an OMNI anycast address into the original IP packet's destination address, and writes the actual destination (along with any additional intermediate hops) into the Segment Routing Header.

Standard IP routing directs the packet to the Client's mobile router entity, where the anycast address identifies the correct OMNI interface for next hop forwarding. When the Client receives the packet, it replaces the IP destination address with the next hop found in the Segment Routing Header and forwards the message via the OMNI interface identified by the anycast address.

Note: The Client need not configure its OMNI interface indexes in one-to-one correspondence with the global OMNI Link-IDs configured for OMNI domain administration since the Client's indexes (i.e., omni0, omni1, omni2, etc.) are used only for its own local interface management.

14.2. Client-Proxy/Server Loop Prevention

After a Proxy/Server has registered an MNP for a Client (see: Section 15), the Proxy/Server will forward all packets destined to an address within the MNP to the Client. The Client will under normal circumstances then forward the packet to the correct destination within its connected (downstream) ENETs.

If at some later time the Client loses state (e.g., after a reboot), it may begin returning packets with destinations corresponding to its MNP to the Proxy/Server as its default router. The Proxy/Server therefore drops any original IP packets received from the Client with a destination address that corresponds to the Client's MNP (i.e., whether ULA or GUA), and drops any carrier packets with both source and destination address corresponding to the same Client's MNP regardless of their origin.

15. Router Discovery and Prefix Registration

Clients engage the MS by sending RS messages with OMNI options under the assumption that one or more Proxy/Server will process the message and respond. The RS message is received by a FHS Proxy/Server, which may in turn forward a proxyed copy of the RS to a Hub Proxy/Server located on the same or different SRT segment. The Hub Proxy/Server then returns an RA message either directly to the Client or via an FHS Proxy/Server acting as a proxy.

Clients and FHS Proxy/Servers include an authentication signature in their RS/RA exchanges when necessary; otherwise, they calculate and include a valid IPv6 ND message checksum (see: Section 12 and Appendix B). FHS and Hub Proxy/Server RS/RA message exchanges over the SRT secured spanning tree instead always include the checksum and omit the authentication signature. Clients and Proxy/Servers use the information included in RS/RA messages to establish NCE state and OMNI link autoconfiguration information as discussed in this section.

For each underlay interface, the Client sends RS messages with OMNI options to coordinate with a (potentially) different FHS Proxy/Server for each interface but with a single Hub Proxy/Server. All Proxy/Servers are identified by their ULA-RNDs and accept carrier packets addressed to their anycast/unicast L2ADDRs; the Hub Proxy/Server may be chosen among any of the Client's FHS Proxy/Servers or may be any other Proxy/Server for the OMNI link. Example ULA/L2ADDR discovery methods are given in [RFC5214] and include data link login parameters, name service lookups, static configuration, a static "hosts" file, etc. In the absence of other information, the Client can resolve the DNS Fully-Qualified Domain Name (FQDN) "linkupnetworks.[domainname]" where "linkupnetworks" is a constant text string and "[domainname]" is a DNS suffix for the OMNI link (e.g., "example.com"). The name resolution will retain a set of DNS resource records with the addresses of Proxy/Servers for the domain.

Each FHS Proxy/Server configures a ULA-RND based on a /64 ULA prefix for the link/segment with randomly-generated Global ID to assure global uniqueness then administratively assigned to FHS Proxy/Servers for the link to assure global consistency. The Client can then configure ULA-MNPs derived from the 64-bit ULA prefix assigned to a FHS Proxy/Server for each underlay interface. The FHS Proxy/Servers discovered over multiple of the Client's underlay interfaces may configure the same or different ULA prefixes, and the Client's ULA-MNP for each underlay interface will fall within the ULA (multilink) subnet relative to each FHS Proxy/Server.

Clients configure OMNI interfaces that observe the properties discussed in previous sections. The OMNI interface and its underlay 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 underlay interfaces. When a first underlay interface transitions to UP, the OMNI interface also transitions to UP. When all underlay interfaces transition to DOWN, the OMNI interface also transitions to DOWN.

When a Client OMNI interface transitions to UP, it sends RS messages to register its MNP and an initial set of underlay interfaces that are also UP. The Client sends additional RS messages to refresh lifetimes and to register/deregister underlay interfaces as they transition to UP or DOWN. The Client's OMNI interface sends initial RS messages over an UP underlay interface with its XLA-MNP as the source (or with a TLA-RND as the source if it does not yet have an MNP) and with destination set to link-scoped All-Routers multicast or the ULA of a specific (Hub) Proxy/Server. The OMNI interface includes an OMNI option per Section 12 with an OMNI Neighbor Coordination sub-option with (Preflen assertion, N/A/U flags and

Window Synchronization parameters), an Interface Attributes sub-option for the underlay interface, a DHCPv6 Solicit sub-option if necessary, and with any other necessary OMNI sub-options such as authentication, Proxy/Server Departure, etc.

The Client then calculates the authentication signature or checksum and prepares to forward the RS over the underlay interface using OAL encapsulation and fragmentation if necessary. If the Client uses OAL encapsulation for RS messages sent to an unsynchronized FHS Proxy/Server over an INET interface, the entire RS message must fit within a single carrier packet (i.e., an atomic fragment) so that the FHS Proxy/Server can verify the authentication signature without having to first reassemble. The OMNI interface selects an Identification value (see: Section 6.6), sets the OAL source address to the ULA-MNP corresponding to the RS source if known (otherwise to a TLA-RND), sets the OAL destination to an OMNI IPv6 anycast address or a known Proxy/Server ULA, optionally includes a Nonce and/or Timestamp, then performs fragmentation if necessary. When L2 encapsulation is used, the Client includes the discovered FHS Proxy/Server L2ADDR or an anycast address as the L2 destination then forwards the resulting carrier packet(s) into the underlay network. Note that the Client does not yet create a NCE, but instead caches the Identification, Nonce and/or Timestamp values included in its RS message transmissions to match against any received RA messages.

When an FHS Proxy/Server receives the carrier packets containing an RS it sets aside the L2 headers, verifies the Identifications and reassembles if necessary, sets aside the OAL header, then verifies the RS authentication signature or checksum. The FHS Proxy/Server then creates/updates a NCE indexed by the Client's RS source address and caches the OMNI Interface Attributes and any Traffic Selector sub-options while also caching the L2 (UDP/IP) and OAL source and destination address information. The FHS Proxy/Server next caches the OMNI Neighbor Coordination sub-option Window Synchronization parameters and N flag to determine its role in processing NS(NUD) messages (see: Section 12.1) then examines the RS destination address. If the destination matches its own ULA, the FHS Proxy/Server assumes the Hub role and acts as the sole entry point for injecting the Client's XLA-MNP into the OMNI link routing system (i.e., after performing any necessary prefix delegation operations) while setting the prefix to fd00::/64 and suffix to the 64-bit MNP, then including a prefix length set to the MNP prefix length plus 64. (For example, if the MNP prefix length is 48, the prefix length field encodes the value 112.) The FHS/Hub Proxy/Server then caches the OMNI Neighbor Coordination sub-option A/U flags to determine its role in processing NS(AR) messages and generating uNA messages (see: Section 12.1).

The FHS/Hub Proxy/Server then prepares to return an RA message directly to the Client by first populating the Cur Hop Limit, Flags, Router Lifetime, Reachable Time and Retrans Timer fields with values appropriate for the OMNI link. The FHS/Hub Proxy/Server next includes as the first RA message option an OMNI option with a neighbor coordination sub-option with Window Synchronization information, an authentication sub-option if necessary and a (proxyed) copy of the Client's original Interface Attributes sub-option with its INET-facing interface information written in the FMT, SRT and LHS Proxy/Server ULA/L2ADDR fields. If the FHS/Hub Proxy/Server's Client-facing interface is different than its INET-facing interface, the Proxy/Server next includes a second Interface Attributes sub-option with ifIndex set to '0' and with a unicast L2 address for its Client-facing interface in the L2ADDR field.

The FHS/Hub Proxy/Server next includes an Origin Indication sub-option that includes the RS L2 source L2ADDR information (see: Section 12.2.16.1), then includes any other necessary OMNI sub-options (either within the same OMNI option or in additional OMNI options). Following the OMNI option(s), the FHS/Hub Proxy/Server next includes any other necessary RA options such as PIOs with (A; L=0) that include the OMNI link MSPs [RFC8028], RIOs [RFC4191] with more-specific routes, Nonce and Timestamp options, etc. The FHS/Hub Proxy/Server then sets the RA source address to its own ULA and destination address to the Client's ULA-MNP (i.e., relative to the ULA /64 prefix for its Client-facing underlay interface) while also recording the corresponding XLA-MNP as an (alternate) index to the Client NCE, then calculates the authentication signature or checksum. The FHS/Hub Proxy/Server finally performs OAL encapsulation with source set to its own ULA and destination set to the OAL source that appeared in the RS, then calculates the OAL checksum, selects an appropriate Identification, fragments if necessary, encapsulates each fragment in appropriate L2 headers with source and destination address information reversed from the RS L2 information and returns the resulting carrier packets to the Client over the same underlay interface the RS arrived on.

When an FHS Proxy/Server receives an RS with a valid authentication signature or checksum and with destination set to link-scoped All-Routers multicast, it can either assume the Hub role itself the same as above or act as a proxy and select the ULA of another Proxy/Server to serve as the Hub. When an FHS Proxy/Server assumes the proxy role or receives an RS with destination set to the ULA of another Proxy/Server, it forwards the message while acting as a proxy. The FHS Proxy/Server creates/updates a NCE for the Client (i.e., based on the RS source address) and caches the OAL source, Window Synchronization, N flag, Interface Attributes addressing information as above then writes its own INET-facing FMT, SRT and LHS Proxy/Server ULA/L2ADDR information into the appropriate

Interface Attributes sub-option fields. The FHS Proxy/Server then calculates and includes the checksum, performs OAL encapsulation with source set to its own ULA and destination set to the ULA of the Hub Proxy/Server, calculates the OAL checksum, selects an appropriate Identification, fragments if necessary, encapsulates each fragment in appropriate L2 headers and sends the resulting carrier packets into the SRT secured spanning tree.

When the Hub Proxy/Server receives the carrier packets, it discards the L2 headers, reassembles if necessary to obtain the proxyed RS, verifies checksums, then performs DHCPv6 Prefix Delegation (PD) to obtain the Client's MNP if the RS source is a (TLA,XLA}-RND. The Hub Proxy/Server then creates/updates a NCE for the Client's XLA-MNP and caches any state (including the A/U flags, OAL addresses, Interface Attributes information and Traffic Selectors), then finally performs routing protocol injection. The Hub Proxy/Server then returns an RA that echoes the Client's (proxyed) Interface Attributes sub-option and with any RA parameters the same as specified for the FHS/Hub Proxy/Server case above. The Hub Proxy/Server then sets the RA source address to its own ULA and destination address to the RS source address; if the RS source address is a TLA-RND, the Hub Proxy/Server also includes the MNP in a DHCPv6 PD Reply OMNI sub-option. The Hub Proxy/Server next calculates the checksum, then encapsulates the RA as an OAL packet with source set to its own ULA and destination set to the ULA of the FHS Proxy/Server that forwarded the RS. The Hub Proxy/Server finally calculates the OAL checksum, selects an appropriate Identification, fragments if necessary, encapsulates each fragment in appropriate L2 headers and sends the resulting carrier packets into the secured spanning tree.

When the FHS Proxy/Server receives the carrier packets it discards the L2 headers, reassembles if necessary to obtain the RA message, verifies checksums then updates the OMNI interface NCE for the Client and creates/updates a NCE for the Hub. The FHS Proxy/Server then sets the P flag in the RA flags field [RFC4389] and proxys the RA by changing the OAL source to its own ULA, changing the OAL destination to the OAL address found in the Client's NCE, and changing the RA destination address to the ULA-MNP of the Client relative to its own /64 ULA prefix while also recording the corresponding XLA-MNP as an alternate index into the Client NCE. (If the RA destination address was a TLA-RND, the FHS Proxy Server determines the MNP by consulting the DHCPv6 PD Reply message sub-option.) The FHS Proxy/Server next includes Window Synchronization parameters responsive to those in the Client's RS, an Interface Attributes sub-option with ifIndex '0' and with its Client-facing interface unicast L2 address if necessary (see above), an Origin Indication sub-option with the Client's cached L2ADDR and an authentication sub-option if necessary. The FHS Proxy/Server finally selects an Identification value per Section 6.6, calculates the

authentication signature or checksum, fragments if necessary, encapsulates each fragment in L2 headers with addresses taken from the Client's NCE and returns the resulting carrier packets via the same underlay interface over which the RS was received.

When the Client receives the carrier packets, it discards the L2 headers, reassembles if necessary and removes the OAL header to obtain the RA message. The Client next verifies the authentication signature or checksum, then matches the RA message with its previously-sent RS by comparing the RS Sequence Number with the RA Acknowledgement Number and also comparing the Nonce and/or Timestamp values if present. If the values match, the Client then creates/ updates OMNI interface NCEs for both the Hub and FHS Proxy/Server and caches the information in the RA message. In particular, the Client caches the RA source address as the Hub Proxy/Server ULA and uses the OAL source address to configure both an underlay interface-specific ULA for the Hub Proxy/Server and the ULA of this FHS Proxy/ Server. The Client then uses the ULA-MNP in the RA destination address to configure its address within the ULA (multilink) subnet prefix of the FHS Proxy/Server. If the Client has multiple underlay interfaces, it creates additional FHS Proxy/Server NCEs and ULA-MNPs as necessary when it receives RAs over those interfaces (noting that multiple of the Client's underlay interfaces may be serviced by the same or different FHS Proxy/Servers). The Client finally adds the Hub Proxy/Server ULA to the default router list if necessary.

For each underlay interface, the Client next caches the (filled-out) Interface Attributes for its own ifIndex and Origin Indication information that it received in an RA message over that interface so that it can include them in future NS/NA messages to provide neighbors with accurate FMT/SRT/LHS information. (If the message includes an Interface Attributes sub-option with ifIndex '0', the Client also caches the L2ADDR as the underlay network-local unicast address of the FHS Proxy//Server via that underlay interface.) The Client then compares the Origin Indication L2ADDR information with its own underlay interface addresses to determine whether there may be NATs on the path to the FHS Proxy/Server; if the L2ADDR information differs, the Client is behind a NAT and must supply the Origin information in IPv6 ND message exchanges with prospective neighbors on the same SRT segment. The Client finally configures default routes and assigns the OMNI Subnet Router Anycast address corresponding to the MNP (e.g., 2001:db8:1:2::) to the OMNI interface.

Following the initial exchange, the FHS Proxy/Server MAY later send additional periodic and/or event-driven unsolicited RA messages per [RFC4861]. (The unsolicited RAs may be initiated either by the FHS Proxy/Server itself or by the Hub via the FHS as a proxy.) The

Client then continuously manages its underlay interfaces according to their states as follows:

* When an underlay interface transitions to UP, the Client sends an RS over the underlay interface with an OMNI option with sub-options as specified above.

* When an underlay interface transitions to DOWN, the Client sends unsolicited NA messages over any UP underlay interface with an OMNI option containing Interface Attributes sub-options for the DOWN underlay interface with Link set to '0'. The Client sends isolated unsolicited NAs when reliability is not thought to be a concern (e.g., if redundant transmissions are sent on multiple underlay interfaces), or may instead set the PNG flag in the OMNI header to trigger a uNA reply.

* When the Router Lifetime for the Hub Proxy/Server nears expiration, the Client sends an RS over any underlay interface to receive a fresh RA from the Hub. If no RA messages are received over a first underlay interface (i.e., after retrying), the Client marks the underlay interface as DOWN and should attempt to contact the Hub Proxy/Server via a different underlay interface. If the Hub Proxy/Server is unresponsive over additional underlay interfaces, the Client sends an RS message with destination set to the ULA of another Proxy/Server which will then assume the Hub role.

* When all of a Client's underlay interfaces have transitioned to DOWN (or if the prefix registration lifetime expires), the Hub Proxy/Server withdraws the MNP the same as if it had received a message with a release indication.

The Client 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 underlay interface (i.e., even after attempting to contact alternate Proxy/Servers), the Client declares this underlay interface as DOWN. When changing to a new FHS or Hub Proxy/Server, the Client also includes a Proxy/Server Departure OMNI sub-option in new RS messages; the (new) FHS Proxy/Server will in turn send uNA messages to the old FHS and/or Hub Proxy/Server to announce the Client's departure as discussed in [I-D.templin-6man-aero].

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 consistent with the information received from the RAs generated by the Hub Proxy/Server.

Whether the OMNI interface IPv6 ND messaging process is initiated from the receipt of an RS message from the IPv6 layer or independently of the IPv6 layer is an implementation matter. Some implementations may elect to defer the OMNI interface internal RS/RA 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 IPv6 layer RS/RA messaging over the OMNI interface, since the messages are not required to drive the OMNI interface internal RS/RA process. (Note that this same logic applies to IPv4 implementations that employ "ICMP Router Discovery" [RFC1256].)

Note: The Router Lifetime value in RA messages indicates the time before which the Client must send another RS message over this underlay 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). Proxy/Servers are therefore responsible for keeping MS state alive on a shorter timescale than the Client may be required to do on its own behalf.

Note: On certain multicast-capable underlay interfaces, Clients should send periodic unsolicited multicast NA messages and Proxy/Servers 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 Neighbor Unreachability Detection (NUD) exchanges to test reachability.

Note: If a single FHS Proxy/Server services multiple of a Client's underlay interfaces, Window Synchronization will initially be repeated for the RS/RA exchange over each underlay interface, i.e., until the Client discovers the many-to-one relationship. This will naturally result in a single window synchronization that applies over the Client's multiple underlay interfaces for the same FHS Proxy/Server.

Note: Although the Client's FHS Proxy/Server is a first-hop segment node from its own perspective, the Client stores the Proxy/Server's FMT/SRT/ULA/L2ADDR as last-hop segment (LHS) information to supply to neighbors. This allows both the Client and Hub Proxy/Server to supply the information to neighbors that will perceive it as LHS information on the return path to the Client.

Note: The Hub Proxy/Server injects Client XLA-MNP into the OMNI link routing system by simply creating a route-to-interface forwarding table entry for fd00::{MNP}/N via the OMNI interface. The dynamic routing protocol will notice the new entry and propagate the route to its peers. If the Hub receives additional RS messages, it need

not re-create the forwarding table entry (nor disturb the dynamic routing protocol) if an entry is already present. If the Hub ceases to receive RS messages from any of the Client's interfaces, it removes the Client XLA-MNP from the forwarding table (i.e., after a short delay) resulting in its removal also from the routing system.

Note: If the Client's initial RS message includes an anycast L2 destination address, the FHS Proxy/Server returns the solicited RA using the same anycast address as the L2 source while including an Interface Attributes sub-option with ifIndex '0' and its true unicast address in the L2ADDR. When the Client sends additional RS messages, it includes this FHS Proxy/Server unicast address as the L2 destination and the FHS Proxy/Server returns the solicited RA using the same unicast address as the L2 source. This will ensure that RS/RA exchanges are not impeded by any NATs on the path while avoiding long-term exposure of messages that use an anycast address as the source.

Note: The Origin Indication sub-option is included only by the FHS Proxy/Server and not by the Hub (unless the Hub is also serving as an FHS).

Note: Clients should set the N/A/U flags consistently in successive RS messages and only change those settings when an FHS/Hub Proxy/Server service profile update is necessary.

Note: After a Client has discovered its ULA-MNPs for a given set of FHS Proxy/Servers, it should begin using its XLA-MNP as the IPv6 ND message source address and ULA-MNP as the OAL source address in future IPv6 ND messages and refrain from further use of TLAs. In any case, the Client SHOULD NOT gratuitously configure and use large numbers of additional TLAs, as doing so would simply result in address change churn in NCEs with no operational advantages.

Note: Although the Client adds the Hub Proxy/Server ULA to the default router list, it also caches the ULAs of the FHS Proxy/Servers on the path to the Hub over each underlying interface. When the Client needs to send a packet to a default router, it therefore selects an ULA corresponding to the selected interface which directs the packet to an FHS Proxy/Server for that interface. The FHS Proxy/Server then forwards the packet without disturbing the Hub.

15.1. Window Synchronization

In environments where Identification window synchronization is necessary, the RS/RA exchanges discussed above observe the principles specified in Section 6.6. Window synchronization is conducted between the Client and each FHS Proxy/Server used to contact the Hub Proxy/Server, i.e., and not between the Client and

the Hub. This is due to the fact that the Hub Proxy/Server is responsible only for forwarding control and data messages via the secured spanning tree to FHS Proxy/Servers, and is not responsible for forwarding messages directly to the Client under a synchronized window. Also, in the reverse direction the FHS Proxy/Servers handle all default forwarding actions without forwarding Client-initiated data to the Hub.

When a Client needs to perform window synchronization via a new FHS Proxy/Server, it sets the RS source address to its own {TLA,XLA}-MNP (or a {TLA,XLA}-RND) and destination address to the ULA of the Hub Proxy/Server (or to All-Routers multicast in an initial RS), then sets the SYN flag and includes an initial Sequence Number for Window Synchronization. The Client then performs OAL encapsulation using its own ULA-MNP (or the TLA-RND) as the source and the ULA of the FHS Proxy/Server as the destination and includes an Interface Attributes sub-option then forwards the resulting carrier packets to the FHS Proxy/Server. The FHS Proxy/Server then extracts the RS message and caches the Window Synchronization parameters then re-encapsulates with its own ULA as the source and the ULA of the Hub Proxy/Server as the target.

The FHS Proxy/Server then forwards the resulting carrier packets via the secured spanning tree to the Hub Proxy/Server, which updates the Client's Interface Attributes and returns a unicast RA message with source set to its own ULA and destination set to the RS source address and with the Client's Interface Attributes echoed. The Hub Proxy/Server then performs OAL encapsulation using its own ULA as the source and the ULA of the FHS Proxy/Server as the destination, then forwards the carrier packets via the secured spanning tree to the FHS Proxy/Server. The FHS Proxy/Server then proxys the message as discussed in the previous section and includes responsive Window Synchronization information. The FHS Proxy/Server then forwards the message to the Client which updates its window synchronization information for the FHS Proxy/Server as necessary.

Following the initial RS/RA-driven window synchronization, the Client can re-assert new windows with specific FHS Proxy/Servers by performing NS/NA exchanges between its own XLA-MNPs and the ULAs of the FHS Proxy/Servers without having to disturb the Hub.

15.2. Router Discovery in IP Multihop and IPv4-Only Networks

On some *NETs, a Client may be located multiple IP hops away from the nearest OMNI link Proxy/Server. 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.). Example routing protocols optimized for MANET/VANET

operations include [RFC3684] and [RFC5614] which operate according to the link model articulated in [RFC5889] and subnet model articulated in [RFC5942].

A Client located potentially multiple *NET hops away from the nearest Proxy/Server prepares an RS message, sets the source address to its XLA-MNP (or to a TLA-RND if it does not yet have an MNP), and sets the destination to link-scoped All-Routers multicast or the unicast ULA of a Proxy/Server the same as discussed above. The OMNI interface then employs OAL encapsulation, sets the OAL source address to a TLA and sets the OAL destination to an OMNI IPv6 anycast address based on either a native IPv6 or IPv4-Compatible IPv6 prefix (see: Section 10).

For IPv6-enabled *NETs, if the underlay interface does not configure an IPv6 GUA the Client injects the TLA into the IPv6 multihop routing system and forwards the message without further encapsulation. Otherwise, the Client encapsulates the message in UDP/IPv6 L2 headers, sets the source to the underlay interface IPv6 address and sets the destination to the same OMNI IPv6 anycast address. The Client then forwards the message into the IPv6 multihop routing system which conveys it to the nearest Proxy/Server that advertises a matching OMNI IPv6 anycast prefix. If the nearest Proxy/Server is too busy, it should forward (without Proxying) the OAL-encapsulated RS to another nearby Proxy/Server connected to the same IPv6 (multihop) network that also advertises the matching OMNI IPv6 anycast prefix.

For IPv4-only *NETs, the Client encapsulates the RS message in UDP/IPv4 L2 headers, sets the source to the underlay interface IPv4 address and sets the destination to the OMNI IPv4 anycast address. The Client then forwards the message into the IPv4 multihop routing system which conveys it to the nearest Proxy/Server that advertises the corresponding IPv4 prefix. If the nearest Proxy/Server is too busy and/or does not configure the specified OMNI IPv6 anycast address, it should forward (without Proxying) the OAL-encapsulated RS to another nearby Proxy/Server connected to the same IPv4 (multihop) network that configures the OMNI IPv6 anycast address. (In environments where reciprocal RS forwarding cannot be supported, the first Proxy/Server should instead return an RA based on its own MSP(s).)

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 such as a roadside unit or another MANET/VANET node). This process repeats iteratively until the RS message is received by a penultimate *NET hop within single-

hop communications range of a Proxy/Server, which forwards the message to the Proxy/Server.

When a Proxy/Server that configures the OMNI IPv6 anycast OAL destination receives the message, it decapsulates the RS and assumes either the Hub or FHS role (in which case, it forwards the RS to a candidate Hub). The Hub Proxy/Server then prepares an RA message with source address set to its own ULA and destination address set to the RS source address if it is acting only as the Hub (or to the Client ULA-MNP within its ULA subnet prefix if it is also acting as the FHS Proxy/Server). The Hub Proxy/Server then performs OAL encapsulation with the RA OAL source/destination set to the RS OAL destination/source and forwards the RA either to the FHS Proxy/ Server or directly to the Client.

When the Hub or FHS Proxy/Server forwards the RA to the Client, it encapsulates the message in L2 encapsulation headers (if necessary) with (src, dst) set to the (dst, src) of the RS L2 encapsulation headers. The Proxy/Server then forwards the message to a *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 Client, which decapsulates the message and performs autoconfiguration the same as if it had received the RA directly from a Proxy/Server on the same physical link. The Client then injects the ULA-MNP into the IPv6 multihop routing system if necessary, then begins using the ULA-MNP as its OAL source address and suspends use of its TLA since it now has a unique address within the FHS Proxy/Server's "Multilink Subnet".

Note: When the RS message includes anycast OAL and/or L2 encapsulation destinations, the FHS Proxy/Server must use the same anycast addresses as the OAL and/or L2 encapsulation sources to support forwarding of the RA message and any initial data packets over any NATs on the path. When the Client receives the RA, it will discover its unicast ULA-MNP and/or L2 encapsulation addresses and can forward future packets using the unicast (instead of anycast) addresses to populate NAT state in the forward path. (If the Client does not have immediate data to send to the FHS Proxy/Server, it can instead send an OAL "bubble" - see Section 6.10.) After the Client begins using unicast OAL/L2 encapsulation addresses in this way, the FHS Proxy/Server should also begin using the same unicast addresses in the reverse direction.

Note: When an OMNI interface configures a TLA, any nodes that forward an encapsulated RS message with the TLA as the OAL source must not consider the message as being specific to a particular OMNI link. TLAs can therefore also serve as the source and destination addresses of unencapsulated IPv6 data communications within the

local routing region, and if the TLAs are injected into the local network routing protocol their prefix length must be set to 128.

Note: Each node normally conducts the multi-hop relaying between intermediate forwarding nodes using the same underlay interface in both the inbound and outbound directions, i.e. as opposed to different underlay interfaces. The final forwarding node within range of a Proxy/Server could use the same or a different underlay interface to exchange packets with the Proxy/Server, but may not be well positioned to perform multilink selections over multiple underlay interfaces on behalf of multihop dependent peers.

15.3. DHCPv6-based Prefix Registration

When a Client is not pre-provisioned with an MNP (or, when the Client requires additional MNP delegations), it requests the MS to select MNPs on its behalf and set up the correct routing state. The DHCPv6 service [RFC8415] supports this requirement.

When a Client requires the MS to select MNPs, it sends an RS message with source set to a TLA-RND. If the Client requires only a single MNP delegation, it can then include a OMNI Node Identification sub-option plus an OMNI Neighbor Coordination sub-option with Preflen set to the length of the desired MNP. If the Client 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 Client then sets the RS destination to link-scoped All-Routers multicast and sends the message using OAL encapsulation and fragmentation if necessary as discussed above.

When the Hub Proxy/Server receives the RS message, it performs OAL reassembly if necessary. Next, if the RS source is a TLA-RND and/or the OMNI option includes a DHCPv6 message sub-option, the Hub Proxy/Server 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 Hub Proxy/Server generates a DHCPv6 Solicit message on behalf of the Client using an IA_PD option with the prefix length set to the OMNI Neighbor Coordination header Preflen value and with a Client Identifier formed from the OMNI option Node Identification sub-option; otherwise, the Hub Proxy/Server uses the DHCPv6 Solicit message contained in the OMNI option. The Hub Proxy/Server then sends the DHCPv6 message to the DHCPv6 Server, which delegates MNPs and returns a DHCPv6 Reply message with PD parameters. (If the Hub Proxy/Server wishes to defer creation of Client 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 Hub Proxy/Server packs any state information needed to return an RA to the Client in the Relay-forward Interface ID option so that the information will be echoed back in the Relay-reply.)

When the Hub Proxy/Server receives the DHCPv6 Reply, it creates XLA-MNPs based on the delegated MNPs and creates OMNI interface XLA-MNP forwarding table entries (i.e., to prompt the dynamic routing protocol). The Hub Proxy/Server then sends an RA back to the FHS Proxy/Server with the DHCPv6 Reply message included in an OMNI DHCPv6 message sub-option. The Hub Proxy/Server sets the RA destination address to the RS source address, sets the RA source address to its own ULA, performs OAL encapsulation and fragmentation, performs L2 encapsulation and sends the RA to the Client via the FHS Proxy/Server as discussed above.

When the FHS Proxy/Server receives the RA, it changes the RA destination address to the ULA-MNP for the Client within its own ULA subnet prefix then forwards the RA to the Client. When the Client 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 or DHCPv6-delegated MNP to automatically configure its primary ULA-MNP. The Client will then use these primary MNP-based addresses as the source address of any IPv6 ND messages it sends as long as it retains ownership of the MNP.

Note: when the Hub Proxy/Server is also the FHS Proxy/Server, it forwards the RA message directly to the Client with the destination set to the Client's ULA-MNP (i.e., instead of forwarding via another Proxy/Server).

15.4. OMNI Link Extension

Clients can provide an OMNI link ingress point for other nodes on their (downstream) ENETs that also act as Clients. When Client A has already coordinated with an (upstream) ANET/INET Proxy/Server, Client B on an ENET serviced by Client A can send OAL-encapsulated RS messages with addresses set the same as specified in Section 15.2. When Client A receives the RS message, it infers from the OAL encapsulation that Client B is seeking to establish itself as a Client instead of just a simple ENET Host.

Client A then returns an RA message the same as a Proxy/Server would do as specified in Section 15.2 except that it instead uses its own ULA-MNP as the RA and OAL source addresses and performs (recursive) DHCPv6 Prefix Delegation. The MNP delegation in the RA message must be a sub-MNP from the MNP delegated to Client A. For example, if Client A receives the MNP 2001:db8:1000::/48 it can provide a sub-

delegation such as 2001:db8:1000:2000::/56 to Client B. Client B can in turn sub-delegate 2001:db8:1000:2000::/56 to its own ENET(s), where there may be a further prospective Client C that would in turn request OMNI link services via Client B.

To support this Client-to-Client chaining, Clients send IPv6 ND messages addressed to the OMNI link anycast address via their ANET/INET (i.e., upstream) interfaces, but advertise the OMNI link anycast address into their ENET (i.e., downstream) networks where there may be further prospective Clients wishing to join the chain. The ENET of the upstream Client is therefore seen as an ANET by downstream Clients, and the upstream Client is seen as a Proxy/Server by downstream Clients.

16. Secure Redirection

If the underlay network link model is multiple access, the FHS Proxy/Server is responsible for assuring that address duplication cannot corrupt the neighbor caches of other nodes on the link. When the Client sends an RS message on a multiple access underlay network, the Proxy/Server verifies that the Client is authorized to use the address and responds with an RA (or forwards the RS to the Hub) only if the Client is authorized.

After verifying Client authorization and returning an RA, the Proxy/Server MAY return IPv6 ND Redirect messages to direct Clients located on the same underlay network to exchange packets directly without transiting the Proxy/Server. In that case, the Clients can exchange packets according to their unicast L2 addresses discovered from the Redirect message instead of using the dogleg path through the Proxy/Server. In some underlay networks, however, such direct communications may be undesirable and continued use of the dogleg path through the Proxy/Server may provide better performance. In that case, the Proxy/Server can refrain from sending Redirects, and/or Clients can ignore them.

17. Proxy/Server Resilience

*NETs SHOULD deploy Proxy/Servers in Virtual Router Redundancy Protocol (VRRP) [RFC5798] configurations so that service continuity is maintained even if one or more Proxy/Servers fail. Using VRRP, the Client is unaware which of the (redundant) FHS Proxy/Servers 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.

Proxy/Servers 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 Proxy/Server Failures

In environments where fast recovery from Proxy/Server failure is required, FHS Proxy/Servers SHOULD use proactive Neighbor Unreachability Detection (NUD) in a manner that parallels Bidirectional Forwarding Detection (BFD) [RFC5880] to track Hub Proxy/Server reachability. FHS Proxy/Servers 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 links such as aeronautical radios) and can therefore be tuned for rapid response.

FHS Proxy/Servers perform proactive NUD for Hub Proxy/Servers for which there are currently active Clients. If a Hub Proxy/Server fails, the FHS Proxy/Server can quickly inform Clients of the outage by sending multicast RA messages. The FHS Proxy/Server sends RA messages to Clients with source set to the ULA of the Hub, with destination address set to All-Nodes multicast (ff02::1) [RFC4291] and with Router Lifetime set to 0.

The FHS Proxy/Server SHOULD send MAX_FINAL_RTR_ADVERTISEMENTS RA messages separated by small delays [RFC4861]. Any Clients that have been using the (now defunct) Hub Proxy/Server will receive the RA messages.

19. Transition Considerations

When a Client connects to an *NET link for the first time, it sends an RS message with an OMNI option. If the first hop router recognizes the option, it responds according to the appropriate FHS/Hub Proxy/Server role resulting in an RA message with an OMNI option returned to the Client. The Client then engages this FHS Proxy/Sever according to the OMNI link model specified above. If the first hop router 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 Client 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 Client sends an RS message on a multiple access *NET link with an OMNI option, first hop routers that

recognize the option ensure that the Client is authorized to use the address and return an RA with a non-zero Router Lifetime only if the Client is authorized. First hop routers that do not recognize the OMNI option instead return an RA that makes no statement about the Client's authorization to use the source address. In that case, the Client 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 Client-Proxy/Server communications is through link-layer address mappings as discussed in Appendix D. This arrangement imparts a (virtual) point-to-point link model over the (physical) multiple access link.

20. OMNI Interfaces on Open Internetworks

Client OMNI interfaces configured over IPv6-enabled underlay interfaces on an open Internetwork without an OMNI-aware first-hop router receive IPv6 RA messages with no OMNI options, while OMNI interfaces configured over IPv4-only underlay interfaces receive no IPv6 RA messages at all (but may receive IPv4 RA messages [RFC1256]). Client OMNI interfaces that receive RA messages with OMNI options configure addresses, on-link prefixes, etc. on the underlay interface that received the RA according to standard IPv6 ND and address resolution conventions [RFC4861] [RFC4862]. Client OMNI interfaces configured over IPv4-only underlay interfaces configure IPv4 address information on the underlay interfaces using mechanisms such as DHCPv4 [RFC2131].

Client OMNI interfaces configured over underlay interfaces connected to open Internetworks can apply security services such as VPNs to connect to a Proxy/Server, or can establish a direct link to the Proxy/Server through some other means (see Section 4). In environments where an explicit VPN or direct link may be impractical or undesirable, Client OMNI interfaces can instead send IPv6 ND messages with OMNI options that include authentication signatures.

OMNI interfaces use UDP/IP as L2 encapsulation headers for transmission over open Internetworks with UDP service port number 8060 (see: Section 25.13 and Section 3.6 of [I-D.templin-6man-aero]) for both IPv4 and IPv6 underlay interfaces. The OMNI interface submits original IP packets for OAL encapsulation, then encapsulates the resulting OAL fragments in UDP/IP L2 headers to form carrier packets. (The first four bits following the UDP header determine whether the OAL headers are uncompressed/compressed as discussed in Section 6.4.) The OMNI interface sets the UDP length to the encapsulated OAL fragment length and sets the IP length to an appropriate value at least as large as the UDP datagram.

For Client-Proxy/Server (e.g., "Vehicle-to-Infrastructure (V2I)") neighbor exchanges, the source must include an OMNI option with an authentication sub-option in all IPv6 ND messages. The source can apply HIP security services per [RFC7401] using the IPv6 ND message OMNI option as a "shipping container" to convey an authentication signature in a (unidirectional) HIP "Notify" message. For Client-Client (e.g., "Vehicle-to-Vehicle (V2V)") neighbor exchanges, two Clients can exchange HIP "Initiator/Responder" messages coded in OMNI options of multiple IPv6 NS/NA messages for mutual authentication according to the HIP protocol. (Note: a simple Hashed Message Authentication Code (HMAC) such as specified in [RFC4380] or the QUIC-TLS connection-oriented service [RFC9000] can be used as an alternate authentication service in some environments.)

When an OMNI interface includes an authentication sub-option, it must appear as the first sub-option of the first OMNI option in the IPv6 ND message which must appear immediately following the IPv6 ND message header. When an OMNI interface prepares a HIP message sub-option, it includes its own (H)HIT as the Sender's HIT and the neighbor's (H)HIT if known as the Receiver's HIT (otherwise 0). If (H)HITs are not available within the OMNI operational environment, the source can instead include other IPv6 address types instead of (H)HITs as long as the Sender and Receiver have some way to associate information embedded in the IPv6 address with the neighbor; such information could include a node identifier, vehicle identifier, MAC address, etc.

Before calculating the authentication signature, the source includes any other necessary sub-options (such as Interface Attributes and Origin Indication) and sets both the IPv6 ND message Checksum and authentication signature fields to 0. The source then calculates the authentication signature over the full length of the IPv6 ND message beginning with a pseudo-header of the IPv6 header (i.e., the same as specified in [RFC4443]) and extending over the length of the message. (If the IPv6 ND message is part of an OAL super-packet, the source instead calculates the authentication signature over the remainder of the super-packet.) The source next writes the authentication signature into the sub-option signature field and forwards the message.

After establishing a VPN or preparing for UDP/IP encapsulation, OMNI interfaces send RS/RA messages for Client-Proxy/Server coordination (see: Section 15) and NS/NA messages for route optimization, window synchronization and mobility management (see: [I-D.templin-6man-aero]). These control plane messages must be authenticated while other control and data plane messages are delivered the same as for ordinary best-effort traffic with source address and/or Identification window-based data origin verification. Upper layer protocol sessions over OMNI interfaces that connect over open

Internetworks without an explicit VPN should therefore employ transport- or higher-layer security to ensure authentication, integrity and/or confidentiality.

Clients should avoid using INET Proxy/Servers as general-purpose routers for steady streams of carrier packets that do not require authentication. Clients should therefore perform route optimization to coordinate with other INET nodes that can provide forwarding services (or preferably coordinate directly with peer Clients directly) instead of burdening the Proxy/Server. Procedures for coordinating with peer Clients and discovering INET nodes that can provide better forwarding services are discussed in [I-D.templin-6man-aero].

Clients that attempt to contact peers over INET underlay interfaces often encounter NATs in the path. OMNI interfaces accommodate NAT traversal using UDP/IP encapsulation and the mechanisms discussed in [I-D.templin-6man-aero]. FHS Proxy/Servers include Origin Indications in RA messages to allow Clients to detect the presence of NATs.

Note: Following the initial IPv6 ND message exchange, OMNI interfaces configured over INET underlay interfaces maintain neighbor relationships by transmitting periodic IPv6 ND messages with OMNI options that include HIP "Update" and/or "Notify" messages. When HMAC authentication is used instead of HIP, the Client and Proxy/Server exchange all IPv6 ND messages with HMAC signatures included based on a shared-secret. When QUIC-TLS connections are used, the Client and Proxy/Server observe QUIC-TLS conventions [RFC9000][RFC9001].

Note: OMNI interfaces configured over INET underlay interfaces should employ the Identification window synchronization mechanisms specified in Section 6.6 in order to exclude spurious carrier packets that might otherwise clutter the reassembly cache. This is especially important in environments where carrier packet spoofing and/or corruption is a threat.

Note: NATs may be present on the path from a Client to its FHS Proxy/Server, but never on the path from the FHS Proxy/Server to the Hub where only INET and/or spanning tree hops occur. Therefore, the FHS Proxy/Server does not communicate Client origin information to the Hub where it would serve no purpose.

21. Time-Varying MNPs

In some use cases, it is desirable, beneficial and efficient for the Client to receive a constant MNP that travels with the Client 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 Client 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 Clients that desire time-varying MNPs to obtain short-lived prefixes to send RS messages with a {TLA,XLA}-RND source address and/or with an OMNI option with DHCPv6 Option sub-options. The Client 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 Clients with automated network renumbering services, but may disrupt persistent sessions that would prefer to use a constant address.

22. (H)HITs and Temporary ULA (TLA)s

Clients 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 TLA. For example, when a Client creates an RS message it can set the source to a (H)HIT and destination to link-scoped All-Routers multicast. The IPv6 ND message includes an OMNI option with a HIP message sub-option, and need not include a Node Identification sub-option if the Client's (H)HIT appears in the HIP message. The Client then encapsulates the message in an IPv6 header with the (H)HIT as the source address. The Client then sends the message as specified in Section 15.2.

When the Hub Proxy/Server receives the RS message, it notes that the source was a (H)HIT, then invokes the DHCPv6 protocol to request an MNP prefix delegation while using the (H)HIT (in the form of a DUID) as the Client Identifier. The Hub Proxy/Server then prepares an RA message with source address set to its own ULA and destination set to the source of the RS message. The Hub Proxy/Server next includes an OMNI option with a HIP message sub-option and any DHCPv6 prefix delegation parameters. The Proxy/Server finally encapsulates the RA in an OAL header with source address set to its own ULA and destination set to the RS OAL source address, then returns the encapsulated RA to the Client either directly or by way of the FHS Proxy/Server as a proxy.

Clients can also use (H)HITs and/or TLAs for direct Client-to-Client communications outside the context of any OMNI link supporting infrastructure. When two Clients encounter one another they can use their (H)HITs and/or TLAs as original IPv6 packet source and destination addresses to support direct communications. Clients can also inject their (H)HITs and/or TLAs into an IPv6 multihop routing protocol to enable multihop communications as discussed in Section

15.2. Clients can further exchange other IPv6 ND messages using their (H)HITs and/or TLAs as source and destination addresses.

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

23. Address Selection

Clients assign LLAs to the OMNI interface, but do not use LLAs as IPv6 ND message source/destination addresses nor for addressing ordinary original IP packets exchanged with OMNI link neighbors.

Clients use ULA-MNPs as source/destination IPv6 addresses in the encapsulation headers of OAL packets and use XLA-MNPs as the IPv6 source addresses of the IPv6 ND messages themselves. Clients use TLAs when an MNP is not available, or as source/destination IPv6 addresses for communications within a MANET/VANET local area. Clients can also use (H)HITs instead of ULAs for local communications when operation outside the context of a specific ULA domain and/or source address attestation is necessary.

Clients 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. Clients can also use MNP-based GUAs within multihop routing regions that are currently disconnected from infrastructure as long as the corresponding ULA-MNPs have been injected into the routing system.

Clients use anycast GUAs as OAL and/or L2 encapsulation destination addresses for RS messages used to discover the nearest FHS Proxy/Server. When the Proxy/Server returns a solicited RA, it must also use the same anycast address as the RA OAL/L2 encapsulation source in order to successfully traverse any NATs in the path. The Client should then immediately transition to using the FHS Proxy/Server's discovered unicast OAL/L2 address as the destination in order to minimize dependence on the Proxy/Server's use of an anycast source address.

24. Error Messages

An OAL destination or intermediate node may need to return ICMPv6-like 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, OAL nodes can instead return error messages as an OMNI ICMPv6 Error sub-option in a secured IPv6 ND uNA message.

25. IANA Considerations

The following IANA actions are requested in accordance with [RFC8126] and [RFC8726]:

25.1. "Protocol Numbers" Registry

The IANA is instructed to allocate an Internet Protocol number TBD1 from the 'protocol numbers' registry for the Overlay Multilink Network Interface (OMNI) protocol. Guidance is found in [RFC5237] (registration procedure is IESG Approval or Standards Action).

25.2. "IEEE 802 Numbers" Registry

During final publication stages, the IESG will be requested to procure an IEEE EtherType value TBD2 for OMNI according to the statement found at https://www.ietf.org/about/groups/iesg/statements/ethertypes/.

Following this procurement, the IANA is instructed to register the value TBD2 in the 'ieee-802-numbers' registry for Overlay Multilink Network Interface (OMNI) encapsulation on Ethernet networks. Guidance is found in [RFC7042] (registration procedure is Expert Review).

25.3. "IPv4 Special-Purpose Address" Registry

The IANA is instructed to assign TBD3/N as an "OMNI IPv4 anycast" address/prefix in the "IPv4 Special-Purpose Address" registry in a similar fashion as for [RFC3068]. The IANA is requested to work with the authors to obtain a TBD3/N public IPv4 prefix, whether through an RIR allocation, a delegation from IANA's "IPv4 Recovered Address Space" registry or through an unspecified third party donation.

25.4. "IPv6 Neighbor Discovery Option Formats" Registry

The IANA is instructed to allocate an official Type number TBD4 from the "IPv6 Neighbor Discovery Option Formats" registry for the OMNI option (registration procedure is RFC required). Implementations set Type to 253 as an interim value [RFC4727].

25.5. "Ethernet Numbers" Registry

The IANA is instructed to allocate one Ethernet unicast address TBD5 (suggested value '00-52-14') in the 'ethernet-numbers' registry

under "IANA Unicast 48-bit MAC Addresses" (registration procedure is Expert Review). The registration should appear as follows:

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

Figure 35: IANA Unicast 48-bit MAC Addresses

25.6. "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 (registration procedure is Standards Action or IESG Approval). 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 36: 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.)

25.7. "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 (registration procedure is RFC required):

Value	Sub-Type name	Reference
0	Pad1	[RFCXXXX]
1	PadN	[RFCXXXX]
2	Neighbor Coordination	[RFCXXXX]
3	Interface Attributes	[RFCXXXX]
4	AERO Forwarding Parameters	[RFCXXXX]
5	Traffic Selector	[RFCXXXX]
6	Geo Coordinates	[RFCXXXX]
7	DHCPv6 Message	[RFCXXXX]
8	HIP Message	[RFCXXXX]
9	PIM-SM Message	[RFCXXXX]
10	Fragmentation Report	[RFCXXXX]
11	Node Identification	[RFCXXXX]
12	ICMPv6 Error	[RFCXXXX]
13	QUIC-TLS Message	[RFCXXXX]
14	Proxy/Server Departure	[RFCXXXX]
15-29	Unassigned	
30	Sub-Type Extension	[RFCXXXX]
31	Reserved by IANA	[RFCXXXX]

Figure 37: OMNI Option Sub-Type Values

25.8. "OMNI Geo Coordinates Type Values" (New Registry)

The OMNI Geo Coordinates sub-option (see: Section 12.2.7) contains an 8-bit Type field, for which IANA is instructed to create and maintain a new registry entitled "OMNI Geo Coordinates Type Values". Initial values are given below (registration procedure is RFC required):

Value	Sub-Type name	Reference
0	NULL	[RFCXXXX]
1-252	Unassigned	[RFCXXXX]
253-254	Reserved for Experimentation	[RFCXXXX]
255	Reserved by IANA	[RFCXXXX]

Figure 38: OMNI Geo Coordinates Type

25.9. "OMNI Node Identification ID-Type Values" (New Registry)

The OMNI Node Identification sub-option (see: Section 12.2.12) 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 (registration procedure is RFC required):

Value	Sub-Type name	Reference
0	UUID	[RFCXXXX]
1	HIT	[RFCXXXX]
2	HHIT	[RFCXXXX]
3	Network Access Identifier	[RFCXXXX]
4	FQDN	[RFCXXXX]
5	IPv6 Address	[RFCXXXX]
6-252	Unassigned	[RFCXXXX]
253-254	Reserved for Experimentation	[RFCXXXX]
255	Reserved by IANA	[RFCXXXX]

Figure 39: OMNI Node Identification ID-Type Values

25.10. "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 (registration procedure is RFC required):

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 40: OMNI Option Sub-Type Extension Values

25.11. "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 (registration procedure is RFC required):

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 41: OMNI RFC4380 UDP/IP Header Option

25.12. "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 (registration procedure is RFC required):

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 42: OMNI RFC6081 Trailer Option

25.13. Additional Considerations

The IANA has assigned the UDP port number "8060" for an earlier experimental version of AERO [RFC6706]. This document reclaims the UDP port number "8060" for 'aero' as the service port for UDP/IP encapsulation. (Note that, although [RFC6706] is not widely implemented or deployed, any messages coded to that specification can be easily distinguished and ignored since they include 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) while retaining the existing name 'aero'.

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.)

OMNI interfaces configured over secured ANET/ENET interfaces inherit the physical and/or link-layer security properties (i.e., "protected spectrum") of the connected networks. 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 or undesirable, however, the security services specified in [RFC7401], [RFC4380] or [RFC9000] 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 Proxy/Servers MUST be supported. In one example, the AERO service [I-D.templin-6man-aero] constructs an SRT spanning tree with Proxy/Serves as leaf nodes and secures the spanning tree links with network layer security mechanisms such as IPsec [RFC4301] or WireGuard [WG]. Secured control plane messages are then constrained to travel only over the secured spanning tree paths and are therefore protected from attack or eavesdropping. Other control and data plane messages can travel over route optimized paths that do not strictly follow the secured spanning tree, therefore end-to-end sessions should employ transport- or higher-layer security services. Additionally, the OAL Identification value can provide a first level of data origin authentication to mitigate off-path spoofing in some environments.

Identity-based key verification infrastructure services such as iPSK may be necessary for verifying the identities claimed by Clients. 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 6.12. In environments where spoofing is considered a threat, OMNI nodes SHOULD employ Identification window synchronization and OAL destinations SHOULD configure an (end-system-based) firewall.

27. Implementation Status

AERO/OMNI Release-3.2 was tagged on March 30, 2021, and is undergoing internal testing. Additional internal releases expected within the coming months, with first public release expected end of 1H2021.

Many AERO/OMNI functions are implemented and undergoing final integration. OAL fragmentation/reassembly buffer management code has been cleared for public release.

28. Document Updates

This document does not itself update other RFCs, but suggests that the following could be updated through future IETF initiatives:

* [RFC1191]

* [RFC2675]

* [RFC4291]

* [RFC4443]

* [RFC8201]

Updates can be through, e.g., standards action, the errata process, etc. as appropriate.

29. 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: Amanda Baber, Stuart Card, Donald Eastlake, Adrian Farrel, Michael Matyas, Robert Moskowitz, Madhu Niraula, Greg Saccone, Stephane Tamalet, Eliot Lear, Eduard Vasilenko, Eric Vyncke. Pavel Drasil, Zdenek Jaron and Michal Skorepa are especially recognized for their many helpful ideas and suggestions. Akash Agarwal, Madhuri Madhava Badgandi, Sean Dickson, Don Dillenburg, Joe Dudkowski, Vijayasarathy Rajagopalan, Ron Sackman, Bhargava Raman Sai Prakash 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 in the 1990s, with insights from colleagues including Ron Bonica, Brian Carpenter, Ralph Droms, Christian Huitema, Thomas Narten, Dave Thaler, Joe Touch, Pascal Thubert, and many others who deserve recognition.

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

Throughout the 1990's and into the 2000's, many colleagues supported and encouraged continuation of the work. Beginning with the DEC Project Sequoia effort at the University of California, Berkeley, then moving to the DEC research lab offices in Palo Alto CA, then to Sterling Software at the NASA Ames Research Center, then to SRI in

Menlo Park, CA, then to Nokia in Mountain View, CA and finally to the Boeing Company in 2005 the work saw continuous advancement through the encouragement of many. Those who offered their support and encouragement are gratefully 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.

30. References

30.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

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

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[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>.

[RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI

10.17487/RFC3971, March 2005, <https://www.rfc-editor.org/info/rfc3971>.

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

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

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

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

[RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, DOI 10.17487/RFC4727, November 2006, <https://www.rfc-editor.org/info/rfc4727>.

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

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

[RFC6088] Tsirtsis, G., Giarreta, G., Soliman, H., and N. Montavont, "Traffic Selectors for Flow Bindings", RFC 6088, DOI 10.17487/RFC6088, January 2011, <https://www.rfc-editor.org/info/rfc6088>.

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

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

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

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

[RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <https://www.rfc-editor.org/info/rfc8415>.

30.2. 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/", 3 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)", 10 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.

[EUI] IEEE, I., "Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID, https://standards.ieee.org/wp-content/uploads/import/documents/tutorials/eui.pdf", 3 August 2017.

[I-D.ietf-drip-rid] Moskowitz, R., Card, S. W., Wiethuechter, A., and A. Gurtov, "DRIP Entity Tag (DET) for Unmanned Aircraft System Remote ID (UAS RID)", Work in Progress, Internet-Draft, draft-ietf-drip-rid-28, 17 May 2022,

<https://www.ietf.org/archive/id/draft-ietf-drip-rid-28.txt>.

[I-D.ietf-intarea-tunnels] Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", Work in Progress, Internet-Draft, draft-ietf-intarea-tunnels-10, 12 September 2019, <https://www.ietf.org/archive/id/draft-ietf-intarea-tunnels-10.txt>.

[I-D.ietf-ipwave-vehicular-networking]
Jeong, J. P., "IPv6 Wireless Access in Vehicular Environments (IPWAVE): Problem Statement and Use Cases", Work in Progress, Internet-Draft, draft-ietf-ipwave-vehicular-networking-29, 19 May 2022, <https://www.ietf.org/archive/id/draft-ietf-ipwave-vehicular-networking-29.txt>.

[I-D.templin-6man-aero]
Templin, F. L., "Automatic Extended Route Optimization (AERO)", Work in Progress, Internet-Draft, draft-templin-6man-aero-51, 17 June 2022, <https://www.ietf.org/archive/id/draft-templin-6man-aero-51.txt>.

[I-D.templin-6man-fragrep]
Templin, F. L., "IPv6 Fragment Retransmission and Path MTU Discovery Soft Errors", Work in Progress, Internet-Draft, draft-templin-6man-fragrep-07, 29 March 2022, <https://www.ietf.org/archive/id/draft-templin-6man-fragrep-07.txt>.

[I-D.templin-6man-lla-type]
Templin, F. L., "The IPv6 Link-Local Address Type Field", Work in Progress, Internet-Draft, draft-templin-6man-lla-type-02, 23 November 2020, <https://www.ietf.org/archive/id/draft-templin-6man-lla-type-02.txt>.

[I-D.templin-intarea-parcels]
Templin, F. L., "IP Parcels", Work in Progress, Internet-Draft, draft-templin-intarea-parcels-10, 29 March 2022, <https://www.ietf.org/archive/id/draft-templin-intarea-parcels-10.txt>.

[IPV4-GUA] Postel, J., "IPv4 Address Space Registry, https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml", 14 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", 14 December 2020.

[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>.

[RFC1146] Zweig, J. and C. Partridge, "TCP alternate checksum options", RFC 1146, DOI 10.17487/RFC1146, March 1990, <https://www.rfc-editor.org/info/rfc1146>.

[RFC1149] Waitzman, D., "Standard for the transmission of IP datagrams on avian carriers", RFC 1149, DOI 10.17487/RFC1149, April 1990, <https://www.rfc-editor.org/info/rfc1149>.

[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>.

[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>.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <https://www.rfc-editor.org/info/rfc2675>.

[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>.

[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>.

[RFC3068] Huitema, C., "An Anycast Prefix for 6to4 Relay Routers", RFC 3068, DOI 10.17487/RFC3068, June 2001, <https://www.rfc-editor.org/info/rfc3068>.

[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>.

[RFC3366] Fairhurst, G. and L. Wood, "Advice to link designers on link Automatic Repeat reQuest (ARQ)", BCP 62, RFC 3366, DOI 10.17487/RFC3366, August 2002, <https://www.rfc-editor.org/info/rfc3366>.

[RFC3684] Ogier, R., Templin, F., and M. Lewis, "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)", RFC 3684, DOI 10.17487/RFC3684, February 2004, <https://www.rfc-editor.org/info/rfc3684>.

[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>.

[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>.

[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>.

[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>.

[RFC5237] Arkko, J. and S. Bradner, "IANA Allocation Guidelines for the Protocol Field", BCP 37, RFC 5237, DOI 10.17487/RFC5237, February 2008, <https://www.rfc-editor.org/info/rfc5237>.

[RFC5558] Templin, F., Ed., "Virtual Enterprise Traversal (VET)", RFC 5558, DOI 10.17487/RFC5558, February 2010, <https://www.rfc-editor.org/info/rfc5558>.

[RFC5614] Ogier, R. and P. Spagnolo, "Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding", RFC 5614, DOI 10.17487/RFC5614, August 2009, <https://www.rfc-editor.org/info/rfc5614>.

[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>.

[RFC5889] Baccelli, E., Ed. and M. Townsley, Ed., "IP Addressing Model in Ad Hoc Networks", RFC 5889, DOI 10.17487/RFC5889, September 2010, <https://www.rfc-editor.org/info/rfc5889>.

[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>.

[RFC6081] Thaler, D., "Teredo Extensions", RFC 6081, DOI 10.17487/RFC6081, January 2011, <https://www.rfc-editor.org/info/rfc6081>.

[RFC6214] Carpenter, B. and R. Hinden, "Adaptation of RFC 1149 for IPv6", RFC 6214, DOI 10.17487/RFC6214, April 2011, <https://www.rfc-editor.org/info/rfc6214>.

[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>.

[RFC6247] Eggert, L., "Moving the Undeployed TCP Extensions RFC 1072, RFC 1106, RFC 1110, RFC 1145, RFC 1146, RFC 1379, RFC 1644, and RFC 1693 to Historic Status", RFC 6247, DOI 10.17487/RFC6247, May 2011, <https://www.rfc-editor.org/info/rfc6247>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <https://www.rfc-editor.org/info/rfc7761>.

[RFC7847] Melia, T., Ed. and S. Gundavelli, Ed., "Logical-Interface Support for IP Hosts with Multi-Access Support", RFC 7847, DOI 10.17487/RFC7847, May 2016, <https://www.rfc-editor.org/info/rfc7847>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8726] Farrel, A., "How Requests for IANA Action Will Be Handled on the Independent Stream", RFC 8726, DOI 10.17487/

RFC8726, November 2020, <https://www.rfc-editor.org/info/rfc8726>.

[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., 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>.

[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>.

[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>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/info/rfc9002>.

[WG] WireGuard, W., "WireGuard, Fast, Modern, Secure VPN Tunnel, https://wireguard.com/", 7 March 2022.

Appendix A. OAL Checksum Algorithm

The OAL Checksum Algorithm adopts the 8-bit Fletcher algorithm specified in Appendix I of [RFC1146] as also analyzed in [CKSUM]. [RFC6247] declared [RFC1146] historic for the reason that the algorithms had never seen widespread use with TCP, however this document adopts the 8-bit Fletcher algorithm for a different

purpose. Quoting from Appendix I of [RFC1146], the OAL Checksum Algorithm proceeds as follows:

> "The 8-bit Fletcher Checksum Algorithm is calculated over a sequence of data octets (call them D[1] through D[N]) by maintaining 2 unsigned 1's-complement 8-bit accumulators A and B whose contents are initially zero, and performing the following loop where i ranges from 1 to N:
>
> A := A + D[i]
>
> B := B + A
>
> It can be shown that at the end of the loop A will contain the 8-bit 1's complement sum of all octets in the datagram, and that B will contain (N)D[1] + (N-1)D[2] + ... + D[N]."

To calculate the OAL checksum, the above algorithm is applied over the N-octet concatenation of the OAL pseudo-header and the encapsulated IP packet or packets. Specifically, the algorithm is first applied over the 40 octets of the OAL pseudo-header as data octets D[1] through D[40], then continues over the entire length of the original IP packet(s) as data octets D[41] through D[N].

Appendix B. IPv6 ND Message Authentication and Integrity

OMNI interface IPv6 ND messages are subject to authentication and integrity checks at multiple levels. When an OMNI interface sends an IPv6 ND message over an INET interface, it includes an authentication sub-option with a valid signature but does not include an IPv6 ND message checksum. The OMNI interface that receives the message verifies the OAL checksum as a first-level integrity check, then verifies the authentication signature (while ignoring the IPv6 ND message checksum) to ensure IPv6 ND message authentication and integrity.

When an OMNI interface sends an IPv6 ND message over an underlay interface connected to a secured network, it omits the authentication sub-option but instead calculates/includes an IPv6 ND message checksum. The OMNI interface that receives the message applies any lower-layer authentication and integrity checks, then verifies both the OAL checksum and the IPv6 ND message checksum. (Note that optimized implementations can verify both the OAL and IPv6 ND message checksums in a single pass over the data.) When an OMNI interface sends IPv6 ND messages to a synchronized neighbor, it includes an authentication sub-option only if authentication is necessary; otherwise, it calculates/includes the IPv6 ND message checksum.

When the OMNI interface calculates the authentication signature or IPv6 ND message checksum, it performs the calculation beginning with a pseudo-header of the IPv6 ND message header and extends over all following OAL packet data. In particular, for OAL super-packets any additional original IP packets included beyond the end of the IPv6 ND message are simply considered as extensions of the IPv6 ND message for the purpose of the calculation.

OAL destinations discard carrier packets with unacceptable Identifications and submit the encapsulated fragments in all others for reassembly. The reassembly algorithm rejects any fragments with unacceptable sizes, offsets, etc. and reassembles all others. Following reassembly, the OAL checksum algorithm provides an integrity assurance layer that compliments any integrity checks already applied by lower layers as well as a first-pass filter for any checks that will be applied later by upper layers.

Appendix C. 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 D. Client-Proxy/Server Isolation Through Link-Layer 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 Client and Proxy/Server only without invoking other nodes on the underlay network. This implies that Client-Proxy/Server control messaging should be isolated and not overheard by other nodes on the link.

To support Client-Proxy/Server isolation on some links, Proxy/Servers can maintain an OMNI-specific unicast link-layer address ("MSADDR"). For Ethernet-compatible links, this specification reserves one Ethernet unicast address TBD5 (see: IANA Considerations). For non-Ethernet statically-addressed links MSADDR is reserved per the assigned numbers authority for the link-layer addressing space. For still other links, MSADDR may be dynamically discovered through other means, e.g., link-layer beacons.

Clients 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 link-layer address. In this way, all of the Client's IPv6 ND messages will be received by Proxy/Servers that are configured to accept packets destined to MSADDR. Note that multiple Proxy/Servers on the link could be configured to accept packets destined to MSADDR, e.g., as a basis for supporting redundancy.

Therefore, Proxy/Servers 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 E. Change Log

<< RFC Editor - remove prior to publication >>

Differences from earlier versions:

*Submit for RFC publication.

Author's Address

Fred L. Templin (editor)
The Boeing Company
P.O. Box 3707
Seattle, WA 98124
United States of America

Email: fltemplin@acm.org