Network Working Group   F. Templin, Ed.
Internet-Draft   The Boeing Company
Intended status: Informational   A. Whyman
Expires: January 3, 2022   MWA Ltd c/o Inmarsat Global Ltd
                                                           July 2, 2021

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

draft-templin-6man-omni-33

Abstract

Mobile network platforms and devices (e.g., aircraft of various configurations, terrestrial vehicles, seagoing vessels, 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 interface specification is presented that enables mobile nodes to coordinate with a network-based mobility service and/or with other mobile node peers. This document specifies the transmission of IP packets over Overlay Multilink Network (OMNI) Interfaces.

Table of Contents

1. Introduction

Mobile network platforms and devices (e.g., aircraft of various configurations, terrestrial vehicles, seagoing vessels, 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 underlying interface data link connections to support the "6M's of modern Internetworking" (see below).

Each Client configures a virtual interface (termed the "Overlay Multilink Network Interface (OMNI)") as a thin layer over its underlying interfaces. The OMNI interface is therefore the only interface abstraction exposed to the IP layer and behaves according to the Non-Broadcast, Multiple Access (NBMA) interface principle, while underlying interfaces appear as link layer communication channels in the architecture. The OMNI interface internally employs the "OMNI Adaptation Layer (OAL)" to ensure that original IP packets are delivered without loss due to size restrictions. The OMNI interface connects to a virtual overlay service known as the "OMNI link". The OMNI link spans one or more Internetworks that may include private-use infrastructures and/or the global public Internet itself.

The Client's OMNI interface interacts with the MS and/or other Clients through IPv6 Neighbor Discovery (ND) control message exchanges [RFC4861]. The MS consists of a distributed set of Proxy/Servers (and other infrastructure elements) that also configure OMNI interfaces. An example MS termed "Automatic Extended Route Optimization (AERO)" appears in [I-D.templin-6man-aero]. In terms of precedence, the AERO specification may provide first-principle insights into a representative mobility service architecture as context for this specification.

Each OMNI interface provides a multilink nexus for exchanging inbound and outbound traffic via the correct underlying interface(s). The IP layer sees the OMNI interface as a point of connection to the OMNI link. Each OMNI link has one or more associated Mobility Service Prefixes (MSPs), which are typically IP Global Unicast Address (GUA) prefixes 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. The Client uses the MNP for numbering downstream-attached End User Networks (EUNs) independently of the access network data links selected for data transport. The Client acts as a mobile router on behalf of its EUNs, and uses OMNI interface control messaging to coordinate with Proxy/Servers and/or other Clients. The Client iterates its control messaging over each of the OMNI interface's underlying interfaces in order to register each interface with the MS (see Section 15).

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 underlying interfaces and provides a nexus for Safety-Based Multilink (SBM) operation. Each OMNI interface within the same OMNI domain configures a common ULA prefix [ULA]::/48, and configures a unique 16-bit Subnet ID '*' to construct the sub-prefix [ULA*]::/64 (see: Section 9). The IP layer applies SBM routing to select a specific OMNI interface, then the selected OMNI interface applies Performance-Based Multilink (PBM) internally to select appropriate underlying 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. In particular, the International Civil Aviation Organization (ICAO) Working Group-I Mobility Subgroup is developing a future Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS) and has issued a liaison statement requesting IETF adoption [ATN] in support of ICAO Document 9896 [ATN-IPS]. The IETF IP Wireless Access in Vehicular Environments (ipwave) working group has further included problem statement and use case analysis for OMNI in a document now in AD evaluation for RFC publication [I-D.ietf-ipwave-vehicular-networking]. Still other communities of interest include AEEC, RTCA Special Committee 228 (SC-228) and NASA programs that examine commercial aviation, Urban Air Mobility (UAM) and Unmanned Air Systems (UAS). Pedestrians with handheld devices represent another large class of potential OMNI users.

OMNI supports the "6M's of modern Internetworking" including:

1. Multilink - a Client's ability to coordinate multiple diverse underlying data links 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 network administrative

domains 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 underlying 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.

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

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. 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 Link-Local Address (LLA), while the Unique-Local Address (ULA) used for encapsulation 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 following terms are defined within the scope of this document:

Client
: a network platform/device mobile router that has one or more distinct upstream data link connections grouped together into one or more logical units. The Client's data link connection parameters can change over time due to, e.g., node mobility, link quality, etc. The Client further connects downstream-attached End User Networks (EUNs).

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

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 supporting infrastructure nodes) for the OMNI link. Specific MS details are out of scope for this document, with an example found in [I-D.templin-6man-aero].

Proxy/Server
: a segment routing topology edge node that provides Clients with a multi-purpose interface to the MS. 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.

Hub Proxy/Server
: a single Proxy/Server selected by the Client that provides a designated router and mobility anchor point service for all of the Client's underlying interfaces. Clients normally select the first FHS Proxy/Server they coordinate with to serve in the Hub role, as

all FHS Proxy/Servers are equally capable candidates to serve in that capacity.

First-Hop Segment (FHS) Proxy/Server
a Proxy/Server for an underlying interface of the source Client that forwards packets sent by the source Client over that interface into the segment routing topology. FHS Proxy/Servers act as intermediate forwarding nodes to facilitate RS/RA exchanges between a Client and its Hub Proxy/Server.

Last-Hop Segment (LHS) Proxy/Server
a Proxy/Server for an underlying interface of the target Client that forwards packets received from the segment routing topology to the target Client over that interface.

Segment Routing Topology (SRT)
a multinet forwarding region between the FHS Proxy/Server and LHS Proxy/Server. FHS/LHS Proxy/Servers and the SRT span 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 Prefix (MSP)
an aggregated IP Global Unicast Address (GUA) prefix (e.g., 2001:db8::/32, 192.0.2.0/24, etc.) assigned to the OMNI link and from which more-specific Mobile Network Prefixes (MNPs) are delegated. OMNI link administrators typically obtain MSPs from an Internet address registry, however private-use prefixes can alternatively be used subject to certain limitations (see: Section 10). OMNI links that connect to the global Internet advertise their MSPs to their interdomain routing peers.

Mobile Network Prefix (MNP)
a longer IP prefix delegated from an MSP (e.g., 2001:db8:1000:2000::/56, 192.0.2.8/30, etc.) and assigned to a Client. Clients sub-delegate the MNP to devices located in EUNs. Note that OMNI link Relay nodes may also service non-MNP routes (i.e., GUA prefixes not covered by an MSP) but that these correspond to fixed correspondent nodes and not Clients. Other than this distinction, MNP and non-MNP routes are treated exactly the same by the OMNI routing system.

Access Network (ANET)
a data link service network (e.g., an aviation radio access network, satellite service provider network, cellular operator network, WiFi network, etc.) that connects Clients. 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.

ANET interface
a Client's attachment to a link in an ANET.

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

INET interface
a node's attachment to a link in an INET.

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

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

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

OMNI Adaptation Layer (OAL)
an OMNI interface sublayer service whereby original IP packets admitted into the interface are wrapped in an IPv6 header and subject to fragmentation and reassembly. The OAL is also responsible for generating MTU-related control messages as necessary, and for providing addressing context for OMNI link SRT traversal.

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

OAL packet
an original IP packet encapsulated in OAL headers and trailers, which is then submitted for OAL fragmentation and reassembly.

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

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

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

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

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

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

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

Mobile Network Prefix Link Local Address (MNP-LLA)
an IPv6 Link Local Address that embeds the most significant 64 bits of an MNP in the lower 64 bits of fe80::/64, as specified in Section 8.

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

Administrative Link Local Address (ADM-LLA)
an IPv6 Link Local Address that embeds a 32-bit administratively-assigned identification value in the lower 32 bits of fe80::/96, as specified in Section 8.

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

Multilink
an OMNI interface's manner of managing diverse underlying interface connections to data links as a single logical unit. The OMNI interface provides a single unified interface to upper layers, while underlying interface selections are performed on a per-packet basis considering traffic selectors such as DSCP, flow label, application policy, signal quality, cost, etc. Multilink selections are coordinated in both the outbound and inbound directions.

Multinet
an OAL intermediate node's manner of spanning multiple diverse IP Internetworks and/or private enterprise networks at the OAL layer below IP. Through intermediate node concatenation of SRT bridged network segments, multiple diverse Internetworks (such as the global public IPv4 and IPv6 Internets) can serve as transit segments in a bridged path for forwarding IP packets end-to-end. This bridging capability provide benefits such as supporting IPv4/IPv6 transition and coexistence, joining multiple diverse operator networks into a cooperative single service network, etc.

Multihop
an iterative relaying of IP packets between Client's over an OMNI underlying 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.

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

L3

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

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

Mobility Service Identification (MSID)
Each Proxy/Server is assigned a unique 32-bit Identification (MSID) (see: Section 8). IDs are assigned according to MS-specific guidelines (e.g., see: [I-D.templin-6man-aero]).

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

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

OMNI Domain
The set of all SBM/PBM OMNI links that collectively provides services for a common set of MSPs. Each OMNI domain consists of a set of affiliated OMNI links that all configure the same ::/48 ULA prefix with a unique 16-bit Subnet ID as discussed in Section 9.

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

Multilink Forwarding Vector (MFV)
An MFIB entry that includes soft state for each underlying interface pairwise communication session between peers. MFVs are identified by both a next-hop and previous-hop MFV Index (MFVI), 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.

Multilink Forwarding Vector Index (MVFI)
A 4 octet value selected by an OMNI node when it creates an MFV, then advertised to either a next-hop or previous-hop. OMNI intermediate nodes assign two distinct MFVIs for each MFV and advertise one to the next-hop and the other to the previous-hop. OMNI end systems assign and advertise a single MFVI. See: [I-D.templin-6man-aero] for further discussion.

3. Requirements

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

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

4. Overlay Multilink Network (OMNI) Interface Model

An OMNI interface is a virtual interface configured over one or more underlying interfaces, which may be physical (e.g., an aeronautical radio link, etc.) or virtual (e.g., an Internet or higher-layer "tunnel"). The OMNI interface architectural layering model is the same as in [RFC5558][RFC7847], and augmented as shown in Figure 1. The IP layer therefore sees the OMNI interface as a single L3 interface nexus for multiple underlying interfaces that appear as L2 communication channels in the architecture.

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

Figure 1: OMNI Interface Architectural Layering Model

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

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

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

o VPNed interfaces use security encapsulation over a *NET to a 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.

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

The OMNI interface forwards original IP packets from the network layer (L3) using the OMNI Adaptation Layer (OAL) (see: Section 5) as an encapsulation and fragmentation sublayer service. This "OAL source" then further encapsulates the resulting OAL packets/fragments in *NET headers to create OAL carrier packets for transmission over

underlying interfaces (L2/L1). The target OMNI interface receives the carrier packets from underlying interfaces (L1/L2) and discards the *NET headers. If the resulting OAL packets/fragments are addressed to itself, the OMNI interface acts as an "OAL destination" and performs reassembly if necessary, discards the OAL encapsulation, and delivers the original IP packet to the network layer (L3). If the OAL fragments are addressed to another node, the OMNI interface instead acts as an "OAL intermediate node" by re-encapsulating in new *NET headers and forwarding the new carrier packets over an underlying interface without reassembling or discarding the OAL encapsulation. The OAL source and OAL destination are seen as "neighbors" on the OMNI link, while OAL intermediate nodes provide a virtual bridging service that joins the segments of a (multinet) Segment Routing Topology (SRT).

The OMNI interface can send/receive original IP packets to/from underlying interfaces while including/omitting various encapsulations including OAL, UDP, IP and L2. The network layer can also access the underlying interfaces directly while bypassing the OMNI interface entirely when necessary. This architectural flexibility may be beneficial for underlying interfaces (e.g., some aviation data links) for which encapsulation overhead may be a primary consideration. OMNI interfaces that send original IP packets directly over underlying interfaces without invoking the OAL can only reach peers located on the same OMNI link segment. 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 underlying interfaces are subject to the same path MTU related issues as for any Internetworking path, and do not include per-packet identifications that can be used for data origin verification and/or link-layer retransmissions. Original IP packets presented directly to an underlying interface that exceed the underlying network path MTU are dropped with an ordinary ICMPv6 Packet Too Big (PTB) message returned. These PTB messages are subject to loss [RFC2923] the same as for any non-OMNI IP interface.

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

```
      +------------------------------------------------------------+
      |             Network Layer (Original IP packets)            |
      +--+---------------------------------------------------------+
         |         OMNI Interface (virtual sublayer nexus)         |
         +--------------------------+------------------------------+
                                    |      OAL Encaps/Decaps       |
                                    +------------------------------+
                                    |        OAL Frag/Reass        |
                       +------------+---------------+--------------+
                       | UDP Encaps/Decaps/Compress |
                  +----+---+------------+--------+--+  +--------+
                  | IP E/D |            | IP E/D |     | IP E/D |
              +---+------+-+----+    +--+---+----+     +----+---+--+
              |L2 E/D|   |L2 E/D|    |L2 E/D|               |L2 E/D|
      +-------+------+---+------+----+------+---------------+------+
      |                   Underlying Interfaces                    |
      +------------------------------------------------------------+
```

Figure 2: OMNI Interface Layering

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

o 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 (MNP-LLA) through the algorithmic derivation specified in Section 8 and assign the LLA to the OMNI interface. Since MNP-LLAs are uniquely derived from an MNP, no Duplicate Address Detection (DAD) or Multicast Listener Discovery (MLD) messaging is necessary.

o since Temporary ULAs are statistically unique, they can be used without DAD until an MNP-LLA is obtained.

o underlying 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.

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

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

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

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

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

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

- o L3 sees the OMNI interface as a point of connection to the OMNI link; if there are multiple OMNI links (i.e., multiple MS's), L3 will see multiple OMNI interfaces.

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

Note that even when the OMNI virtual interface is present, applications can still access underlying interfaces either through the network protocol stack using an Internet socket or directly using a raw socket. This allows for intra-network (or point-to-point) communications without invoking the OMNI interface and/or OAL. For example, when an IPv6 OMNI interface is configured over an underlying IPv4 interface, applications can still invoke IPv4 intra-network communications as long as the communicating endpoints are not subject to mobility dynamics.

Figure 3 depicts the architectural model for a source Client with an attached EUN connecting to the OMNI link via multiple independent *NETs. The Client's OMNI interface sends IPv6 ND messages over available underlying interfaces to FHS Proxy/Servers using any necessary *NET 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 message response 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.

```
                           +--------------+        (:::)-.
                           |Source Client |<-->.-(::EUN:::)
                           +--------------+      `-(::::)-'
                           |OMNI interface|
                           +----+----+----+
                  +--------|IF#1|IF#2|IF#n|------ +
                 /         +----+----+----+        \
                /                 |                 \
               /                  |                  \
              v                   v                   v
           (:::)-.              (:::)-.              (:::)-.
      .-(::*NET:::)        .-(::*NET:::)        .-(::*NET:::)
        `-(::::)-'           `-(::::)-'           `-(::::)-'
         +-----+              +-----+              +-----+
    ...  |FHS#1|  .........   |FHS#2|   .........  |FHS#n|  ...
   .     +--|--+              +--|--+              +--|--+     .
   .        |                    |                    |
   .        \                    v                    /        .
   .         \                                       /         .
   .           v                 (:::)-.           v            .
   .                        .-(:::::::)                        .
   .                    .-(::: Segment :::)-.                  .
   .                  (:::::   Routing   ::::)                 .
   .                     `-(:: Topology ::)-'                  .
   .                         `-(:::::::-'                      .
   .                  /          |          \                  .
   .                 /           |           \                 .
   .                v            v            v
   .     +-----+              +-----+              +-----+     .
    ...  |LHS#1|  .........   |LHS#2|   .........  |LHS#m|  ...
         +--|--+              +--|--+              +--|--+
             \                   |                    /
              v                  v                   v
                       <-- Target Clients -->
```

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

After the initial IPv6 ND message exchange, the source Client (and/or any nodes on its attached EUNs) 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 underlying *NETs. The FHS Proxy/Server then forwards them over the SRT which delivers them to an LHS Proxy/Server, and the LHS Proxy/Server in turn forwards the packets to the target Client. (Note that when the

source and target Client are on the same SRT segment, the FHS and LHS Proxy/Servers are often one and the same.)

When a Client coordinates with its FHS Proxy/Servers, it selects one to serve in the Hub Proxy/Server role (not shown in the figure). Clients then register all of their underlying 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 and mobility anchor point service for the Client.

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 an SRT intermediate system themselves (or directly to correspondents in the same SRT segment) to reduce Proxy/Server load.

5. OMNI Interface Maximum Transmission Unit (MTU)

The OMNI interface observes the link nature of tunnels, including the Maximum Transmission Unit (MTU), Maximum Reassembly Unit (MRU) and the role of fragmentation and reassembly [I-D.ietf-intarea-tunnels]. The OMNI interface is configured over one or more underlying interfaces as discussed in Section 4, where the interfaces (and their associated *NET paths) may have diverse MTUs. OMNI interface considerations for accommodating original IP packets of various sizes are discussed in the following sections.

IPv6 underlying interfaces are REQUIRED to configure a minimum MTU of 1280 bytes and a minimum MRU of 1500 bytes [RFC8200]. Therefore, the minimum IPv6 path MTU is 1280 bytes since routers on the path are not permitted to perform network fragmentation even though the destination is required to reassemble more. The network therefore MUST forward original IP packets of at least 1280 bytes without generating an IPv6 Path MTU Discovery (PMTUD) Packet Too Big (PTB) message [RFC8201]. (While the source can apply "source fragmentation" for locally-generated IPv6 packets up to 1500 bytes and larger still if it knows the destination configures a larger MRU, this does not affect the minimum IPv6 path MTU.)

IPv4 underlying interfaces are REQUIRED to configure a minimum MTU of 68 bytes [RFC0791] and a minimum MRU of 576 bytes [RFC0791][RFC1122]. Therefore, when the Don't Fragment (DF) bit in the IPv4 header is set to 0 the minimum IPv4 path MTU is 576 bytes since routers on the path support network fragmentation and the destination is required to

reassemble at least that much. The OMNI interface therefore MUST set DF to 0 in the IPv4 encapsulation headers of carrier packets that are no larger than 576 bytes, and SHOULD set DF to 1 in larger carrier packets unless it has a way to determine the encapsulation destination MRU and has carefully considered the issues discussed in Section 6.10.

The OMNI interface configures an MTU and MRU of 9180 bytes [RFC2492]; the size is therefore not a reflection of the underlying interface or *NET path MTUs, but rather determines the largest original IP packet the OAL (and/or underlying interface) can forward or reassemble. For each OAL destination (i.e., for each OMNI link neighbor), the OAL source may discover "hard" or "soft" Reassembly Limit values smaller than the MRU based on receipt of IPv6 ND messages with OMNI Reassembly Limit sub-options (see: Section 12.2.10). The OMNI interface employs the OAL as an encapsulation sublayer service to transform original IP packets into OAL packets/fragments, and the OAL in turn uses *NET encapsulation to forward carrier packets over the underlying interfaces (see: Section 6).

6. The OMNI Adaptation Layer (OAL)

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

These carrier packets travel over one or more underlying networks spanned by OAL intermediate nodes in the SRT, which re-encapsulate by removing the *NET headers of the first underlying network and appending *NET headers appropriate for the next underlying network in succession. (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 *NET headers and reassembles the resulting OAL fragments into an OAL packet as described in Section 6.3. The OAL destination then decapsulates the OAL packet to obtain the original IP packet, which it then delivers to the network layer. 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 other SRT infrastructure node types such as those 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 underlying 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 inserts an IPv6 encapsulation header but does not decrement the Hop Limit/TTL of the original IP packet since encapsulation occurs at a layer below IP forwarding [RFC2473]. The OAL source copies the "Type of Service/Traffic Class" [RFC2983] and "Congestion Experienced" [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 Hop Limit to a conservative value sufficient to enable loop-free forwarding over multiple concatenated OMNI link segments and sets the Payload Length to the length of the original IP packet.

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

When the OAL source forwards an original IP packet toward a final destination via an ANET underlying interface, it sets the OAL IPv6 header source address to its own ULA and sets the destination to either the Administrative ULA (ADM-ULA) of the ANET peer or the Mobile Network Prefix ULA (MNP-ULA) corresponding to the final destination (see below). The OAL source then fragments the OAL packet if necessary, encapsulates the OAL fragments in any ANET headers and sends the resulting carrier packets to the ANET peer which either reassembles before forwarding if the OAL destination is its own ULA or forwards the fragments toward the true OAL destination without first reassembling otherwise.

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

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

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

Figure 4: OAL Packet Before Fragmentation

The OAL source next selects a 32-bit Identification value for the packet as specified in Section 6.6 then calculates an OAL checksum using the algorithm specified in Appendix A. The OAL source calculates the checksum over the entire OAL packet beginning with a pseudo-header of the IPv6 header similar to that found in Section 8.1 of [RFC8200] and extending to the end of the (0-initialized) checksum trailer. The OAL IPv6 pseudo-header is formed as shown in Figure 5:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      OAL Source Address                       +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                   OAL Destination Address                     +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       OAL Payload Length      |     zero      |  Next Header  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Identification                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: OAL IPv6 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 bytes) as the worst case for OAL fragmentation regardless of the underlying interface IP protocol version since IPv6/IPv4 protocol translation and/or IPv6-in-IPv4 encapsulation may occur in any *NET path. By always assuming the IPv4 minimum even for IPv6 underlying interfaces, the OAL source may produce smaller fragments with additional encapsulation overhead but will always interoperate and never run the risk of loss due to an MTU restriction or due to presenting an underlying interface with a carrier packet that exceeds its MRU. Additionally, the OAL path could traverse multiple SRT segments with intermediate OAL forwarding nodes performing re-encapsulation where the *NET encapsulation of the previous segment is replaced by the *NET encapsulation of the next segment which may be based on a different IP protocol version and/or encapsulation sizes.

The OAL source therefore assumes a default minimum path MTU of 576 bytes at each SRT segment for the purpose of generating OAL fragments for *NET encapsulation and transmission as carrier packets. Each successive SRT intermediate node includes either a 20 byte IPv4 or 40 byte IPv6 header, an 8 byte UDP header and in some cases an IP security encapsulation (40 bytes maximum assumed) during re-encapsulation. Intermediate nodes at any SRT segment may also insert

a Routing Header (assume 40 bytes worst-case) as an extension to the existing 40 byte OAL IPv6 header plus 8 byte Fragment Header. Therefore, assuming a worst case of (40 + 40 + 8) = 88 bytes for *NET encapsulation plus (40 + 40 + 8) = 88 bytes for OAL encapsulation leaves no less than (576 - 88 - 88) = 400 bytes to accommodate a portion of the original IP packet/fragment. The OAL source therefore sets a minimum Maximum Payload Size (MPS) of 400 bytes as the basis for the minimum-sized OAL fragment that can be assured of traversing all 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. (Note that the OAL source includes the 2 octet trailer as part of the payload during fragmentation, and the OAL destination regards it as ordinary payload until reassembly and checksum verification are complete.)

The OAL source SHOULD maintain "path MPS" values for individual OAL destinations initialized to the minimum MPS and increased to larger values (up to the OMNI interface MTU) if better information is known or discovered. For example, when *NET peers share a common underlying link or a fixed path with a known larger MTU, the OAL source can set path MPS to this larger size (i.e., instead of 576 bytes) as long as the *NET peer reassembles before re-encapsulating and forwarding (while re-fragmenting if necessary). Also, if the OAL source has a way of knowing the maximum *NET encapsulation size for all SRT segments along the path it may be able to increase path MPS to reserve additional room for payload data. The OAL source must include the uncompressed OAL header size in its path MPS calculation, since 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 in case the probe was lost). For this purpose, the OAL source can send an NS message with one or more OMNI options with large PadN sub-options (see: Section 12) in order to receive a small NA response from the OAL destination. While observing the minimum MPS will always result in robust and secure behavior, the OAL source should optimize path MPS values when more efficient utilization may result in better performance (e.g. for wireless aviation data links). (If so, the OAL source should maintain separate path MPS values for each (source, target) underlying interface pair for the same OAL destination, since each underlying interface pair may support a different path MPS.)

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 into "atomic fragments" by including a Fragment Header with Fragment Offset and More Fragments both set to 0.

For each fragment produced, the OAL source writes an ordinal number for the fragment into the Reserved field in the IPv6 Fragment Header. Specifically, the OAL source writes the ordinal number '0' for the first fragment, '1' for the second fragment, '2' for the third fragment, etc. up to and including the final fragment. Since the minMPS is 400 and the MTU is 9180, the OAL source will produce at most 23 fragments for each OAL packet; the OAL destination therefore unconditionally discards any fragments with an ordinal number larger than 22.

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

```
        +----------+--+-------------+
        |  OAL Hdr |FH|   Frag #0   |
        +----------+--+-------------+
            +----------+--+-------------+
            |  OAL Hdr |FH|   Frag #1   |
            +----------+--+-------------+
                +----------+--+-------------+
                |  OAL Hdr |FH|   Frag #2   |
                +----------+--+-------------+
                                  ....
                    +----------+--+------------+----+
                    |  OAL Hdr |FH| Frag #(N-1) |Csum|
                    +----------+--+------------+----+
        a) OAL fragments after fragmentation
           (FH = Fragment Header; Csum appears only in final fragment)


        +--------+--+-----+-----+-----+-----+-----+----+
        |OAL Hdr |FH|      Original IP packet     |Csum|
        +--------+--+-----+-----+-----+-----+-----+----+
        b) An OAL atomic fragment with FH but no fragmentation.


        +--------+----------+--+-------------+
        |*NET Hdr|  OAL Hdr |FH|   Frag #i   |
        +--------+----------+--+-------------+
        c) OAL carrier packet after *NET encapsulation
```

Figure 6: OAL Fragments and Carrier Packets

6.2. OAL *NET Encapsulation and Re-Encapsulation

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

The *NET source encapsulates the OAL information immediately following the *NET innermost 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 followed by any IPv6 extension headers followed by upper layer protocol headers and data. Otherwise, the first four bits include a "Type" value, and the OAL information appears in an alternate format as specified in Section 6.4). Alternate formats for Types '0' and '1' are currently specified, while all other Type values except '4' and '6' are reserved for future use.

The OAL node prepares the innermost *NET encapsulation header as follows:

o For UDP, the *NET source sets the UDP source port to 8060 (i.e., the port number reserved for AERO/OMNI). When the *NET destination is a Proxy/Server or Bridge, the *NET source sets the UDP destination port to 8060; otherwise, the *NET source sets the UDP destination port to its cached port number value for the peer. The *NET source finally sets the UDP Length the same as specified in [RFC0768].

o For IP encapsulation, the IP port number is set to TBD1 as the Internet Protocol number for OMNI. For IPv4, the *NET source sets the Total Length the same as specified in [RFC0791]; for IPv6, the *NET source sets the Payload Length the same as specified in [RFC8200].

o For encapsulations over Ethernet-compatible L2s, the EtherType is set to TBD2 as the EtherType number for OMNI. Since the Ethernet header does not include a length field, for the OMNI EtherType the Ethernet header is followed by a two-octet length field followed immediately by the encapsulated OAL information. The length field encodes the length in octets (in network byte order) of the information following the Ethernet header including the length field, but excluding the Ethernet trailer.

When a *NET source includes a UDP header, it SHOULD calculate and include a UDP checksum in carrier packets with full OAL headers to ensure header integrity, and MAY disable UDP checksums in carrier packets with compressed OAL headers. If the *NET 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 *NET destination. If the *NET 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 a *NET source sends carrier packets with compressed OAL headers and with UDP checksums disabled, mis-delivery due to corruption of the 4-octet Multilink Forwarding Vector Index (MFVI) is possible but

unlikely since the corrupted index would somehow have to match valid state in the (sparsely-populated) Multilink Forwarding Information Based (MFIB). In the unlikely event that a match occurs, an OAL destination may receive a mis-delivered carrier packet but can immediately reject the packet if it has 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. Finally, 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 that somehow eluded earlier checks. See: [RFC6935][RFC6936] for further discussion.

For *NET encapsulations over IP, when the *NET source is also the OAL source it next copies the "Type of Service/Traffic Class" [RFC2983] and "Congestion Experienced" [RFC3168] values in the OAL IPv6 header into the corresponding fields in the *NET IP header, then (for IPv6) set the *NET IPv6 header "Flow Label" as specified in [RFC6438]. The *NET source then sets the *NET 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 *NET source decrements the OAL IPv6 header Hop Limit and discards the carrier packet if the value reaches 0. The *NET source then copies the "Type of Service/ Traffic Class" and "Congestion Experienced" values from the previous hop *NET encapsulation header into the OAL IPv6 header, then finally sets the source and destination IP addresses the same as above.

Following *NET encapsulation/re-encapsulation, the *NET source sends the resulting carrier packets over one or more underlying interfaces. The underlying interfaces often connect directly to physical media on the local platform (e.g., a laptop computer with WiFi, etc.), but in some configurations the physical media may be hosted on a separate Local Area Network (LAN) node. In that case, the OMNI interface can establish a Layer-2 VLAN or a point-to-point tunnel (at a layer below the underlying interface) to the node hosting the physical media. The OMNI interface may also apply encapsulation at the underlying interface layer (e.g., as for a tunnel virtual interface) such that carrier packets would appear "double-encapsulated" on the LAN; the node hosting the physical media in turn removes the LAN encapsulation prior to transmission or inserts it following reception. Finally, the underlying interface must monitor the node hosting the physical media (e.g., through periodic keepalives) so that it can convey up/down/status information to the OMNI interface.

6.3. OAL *NET Decapsulation and Reassembly

When an OMNI interface receives a carrier packet from an underlying interface, it discards the *NET encapsulation headers and examines the OAL header of the enclosed OAL fragment. If the OAL fragment is addressed to a different node, the OMNI interface (acting as an OAL intermediate node) re-encapsulates and forwards 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 the ordinal numbers of each accepted fragment of the same OAL packet (i.e., as Frag #0, Frag #1, Frag #2, etc.), then admits the fragments into the reassembly cache. When reassembly is complete, the OAL destination next verifies the OAL packet checksum and discards the packet if the checksum is incorrect. If the OAL packet was accepted, the OAL destination then removes the OAL header/trailer, then delivers the original IP packet to the network layer.

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 the OMNI interface MTU [CRC], but other paths may traverse links (such as tunnels over IPv4) that do not include integrity checks. 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

When OAL source, intermediate and destination nodes exchange IPv6 ND messages to establish header compression state. After an initial IPv6 ND message exchange, OAL nodes can apply OAL Header Compression to significantly reduce encapsulation overhead.

Each node establishes a Multilink Forwarding Information Based (MFIB) soft state entry known as a Multilink Forwarding Vector (MVF) which

supports both carrier packet forwarding and OAL header compression/ decompression. For OAL sources, the MFV is referenced by a single MFV Index (MFVI) that provides compression/decompression context for the next hop. For OAL destinations, the MFV is referenced by a single MFVI that provides context for the previous hop. For OAL intermediate nodes, the MFV is referenced by two MFVIs - one for the previous hop and one for the next hop.

When an OAL node uses *NET encapsulation to forward carrier packets directly to a next hop, it can omit significant portions of the OAL IPv6 header and Fragment Headers while including an OAL compressed header. The full OAL IPv6 header or compressed header follows immediately after the innermost *NET encapsulation (i.e., UDP, IP or L2) as discussed in Section 6.2. Two OAL compressed header types (Type '0' and Type '1') are specified below, while future documents may specify additional types.

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

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ *
   |Type=0 | Traffic Class |           Flow Label                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Next Header  |  Hop Limit  |M|      Identification (0-1)     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       Identification (2-3)    |           MFVI (0-1)          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            MFVI (2-3)         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

The format begins with a 4-bit Type field set to 0, and is followed by the uncompressed Traffic Class and Flow Label copied from the OAL IPv6 header, followed by a Next Header field set to the protocol number for the header immediately following the IPv6 Fragment Header. The Next Header field is then followed by a 7-bit compressed Hop Limit field set to the minimum of 127 and the uncompressed OAL IPv6 Hop Limit value. The Hop Limit is then followed by a compressed Fragment Header beginning with a (M)ore Fragments bit followed by a 4-octet Identification and with all other fields omitted. The compressed Fragment Header is then followed by a 4-octet Multilink Forwarding Vector Index (MFVI).

The uncompressed OAL fragment body is then included immediately following the OCH-0 header, and the *NET header length field is

reduced by the difference in length between the compressed headers and full-length IPv6 and Fragment headers. 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.

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Type=1 | Ordinal |R|M|    Fragment Offset      |     ID (0)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Identification (1-3)              |    MFVI (0)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  MFVI (1-3)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

The format begins with a Type field set to 1 and the IPv6 header is omitted entirely. The Type field is followed by a compressed IPv6 Fragment Header with a 5-bit Ordinal number field, a (R)eserved bit set to 0, and with ((M)ore Fragments/Fragment Offset/Identification) copied from the uncompressed fragment header. The compressed Fragment Header is followed by a 4-octet MFVI the same as for OCH-0.

The uncompressed OAL fragment body is then included immediately following the OCH-1 header, and the *NET header length field is reduced by the difference in length between the compressed headers and full-length IPv6 and Fragment headers. The OCH-1 format applies for non-first fragments only; therefore, Ordinal is set 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.

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 *NET header then examines the first four bits immediately following the *NET header. If the bits contain the value 6, the OAL node processes the remainder as an uncompressed OAL fragment, If the bits contain the value 0 or 1, the OAL node instead processes the remainder of the header as an OCH-0 or OCH-1, respectively.

For OCH-O/1, the OAL node then uses the MFVI to locate the cached MFV. The OAL node uses the MFV to determine the next hop

intermediate OAL node for forwarding. During forwarding, the OAL node changes the MFVI to the cached value for the MVF 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. Since OCH-1 does not include Traffic Class, Flow Label, Next Header or Hop Limit information, the OAL node writes the value 0 into those fields when it reconstructs the full OAL headers. The values will be correctly populated during reassembly after an OAL first fragment with an OCH-0 or uncompressed OAL header arrives.

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.

6.5. Carrier Packet in Carrier Packet Encapsulation

When an OAL source is unable to forward carrier packets directly to an OAL destination without the involved services of an OAL intermediate node, the OAL source must regard the OAL intermediate node as an ingress tunnel endpoint. The OAL source must therefore include a NCE and MFV for the OAL destination while the OAL intermediate node must have a NCE and MFV for the egress tunnel endpoint. This will result in encapsulation when carrier packets sent by the OAL source arrive at the OAL intermediate node.

For example, if the OAL source has an NCE/MFV with MFVI 0x2376a7b5 and Identification 0x12345678 for the OAL destination, and the OAL intermediate node has an NCE/MFV with MFVI 0x692a64fc and Identification 0x98765432 for the egress tunnel endpoint, the OAL source prepares the carrier packet using OCH-0/1 compression with the MFVI and Identification corresponding to the OAL destination but with *NET header information addressed to the next hop toward the OAL intermediate node. When the OAL intermediate node receives the carrier packet, it recognizes the MFVI included by the OAL source and determines the correct egress tunnel endpoint.

The OAL intermediate node then discards the *NET headers from the previous hop and encapsulates the original OCH-0/1 within a second OCH-0/1. The OAL intermediate node then includes *NET 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:

```
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                                       |
      |                                       |
      |                                       |
      |        Carrier packet data            |
      |                                       |
      |                                       |
      |                                       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |          Original OCH-0/1             |
      |    MFVI=0x2376a7b5, Id=0x12345678     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |        Encapsulation OCH-0/1          |
      |    MFVI=0x692a64fc, Id=0x98765432     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |            *NET headers               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Carrier Packet in Carrier Packet Encapsulation

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 were spoofing is not considered a threat, OAL nodes send OAL packets with Identifications beginning with an unpredictable Initial Send Sequence (ISS) value [RFC7739] incremented (modulo 2**32) for each successive OAL packet and may reset ISS to a new unpredictable value at any time. In other environments, OMNI interfaces should maintain explicit per-neighbor send and receive windows to exclude spurious carrier packets that might clutter the reassembly cache. 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 frequently 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 neighbor cache entries (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 LLA, which must also match the ULA used for OAL encapsulation. 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 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 NS/RS message with the SYN flag set and with Window set to M 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 a solicited NA/RA ACK response (retransmitting up to MAX_UNICAST_SOLICIT times if necessary).

When OAL B receives the carrier packets containing the NS/RS 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 a solicited NA/RA 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 OAL A's IRS as the Identification for OAL encapsulation then sends the resulting OAL packet to OAL A.

When OAL A receives the carrier packets containing the solicited NA/RA, it notes that their Identification 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 NS/RS SYN message 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 solicited NA/RA, 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. IPv6 ND messages used for window synchronization must therefore fit within a single carrier packet (i.e., within current MPS constraints), since the carrier packets of fragmented IPv6 ND messages with out-of-window Identification values could be part of a DoS attack and should not be admitted into the reassembly cache. OAL B discards all other carrier packets received from OAL A with out-of-window Identifications.

OMNI interface neighbors can employ asymmetric window synchronization as described above using two independent [(NS/RS SYN) -> (NA/RA 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:

- o OAL A prepares an NS/RS SYN message 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 a solicited NA/RA ACK response (retransmitting up to MAX_UNICAST_SOLICIT times if necessary).

- o OAL B receives the carrier packets containing the NS/RS 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 solicited NA/RA 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 NS 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 carrier packets containing the NA/RA 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 NA 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 NA 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 NS/RS 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 an NS/RS SYN with a new unpredictable ISS. When OAL B

receives the NS/RS SYN, it resets its RCV variables and may optionally return either an asymmetric NA/RA ACK or a symmetric NA/RA SYN/ACK to also assert a new ISS. While sending IPv6 ND 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 in IPv6 ND advertisement messages when a reachability confirmation is needed. (OMNI interfaces therefore most often set the PNG flag in (unsolicited) advertisement messages and ignore it in solicitation messages.) When an OMNI interface receives a PNG, it returns a solicited NA ACK with the PNG message Identification in the Acknowledgment, but without updating RCV state variables. OMNI interfaces return unicast solicited NA ACKs even for multicast PNG destination addresses, since OMNI link multicast is based on unicast emulation. OMNI interfaces may also send unsolicited NA messages to request selective retransmissions (see: Section 12.2.11).

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

- OMNI interfaces MUST select new unpredictable ISS values that are outside of the current SND.WND.

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

- OMNI interfaces that receive NA/RA messages with the PNG and/or SYN flag set MUST NOT set the PNG and/or SYN flag in solicited NA responses.

- OMNI interfaces that send NA/RA messages with the PNG and/or SYN flag set MUST ignore solicited NA responses with the PNG and/or SYN flag set.

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

When an OMNI interface sends an RS SYN to the All-Routers multicast address, it may receive multiple unicast RA ACK or SYN/ACK replies - each with a distinct LLA source address. The OMNI interface then creates a separate NCE for each distinct neighbor and completes window synchronization through independent message exchanges with each neighbor. The fact that all neighbors receive the same ISS in the original RS SYN is not a matter for concern, as further window synchronization will be conducted on a per-neighbor basis.

Note: Although OMNI interfaces employ TCP-like window synchronization and support solicited NA ACK responses to NA/RA 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 ULA source address, which also determines the carrier packet Identification window. However, IPv6 ND messages may contain an LLA source address that does not match the ULA 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) underlying interface pairs that exchange carrier packets. Each interface pair represents a distinct underlying network path, and the set of paths traversed may be highly diverse when multiple interface pairs are used. OMNI intermediate nodes therefore SHOULD NOT take actions based on 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 sub-options that include a list of (Identification, Bitmap)-tuples for fragments received and missing from this OAL source (see: Section 12). 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 Fragmentation Report. 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 same OAL packet 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 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.

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.

Ordinary PTB messages with ICMPv4 header "unused" field or ICMPv6 header Code field value 0 are hard errors that always indicate that a packet has been dropped due to a real MTU restriction. In particular, the OAL source drops the packet and returns a PTB hard error if the packet exceeds the OAL destination MRU. However, the OMNI interface can also forward large original IP packets via OAL encapsulation and fragmentation while at the same time returning PTB

soft error messages (subject to rate limiting) if it deems the original IP packet too large according to factors such as link performance characteristics, reassembly congestion, etc. This ensures that the path MTU is adaptive and reflects the current path used for a given data flow. The OMNI interface can therefore continuously forward packets without loss while returning PTB soft error messages recommending a smaller size if necessary. Original sources that receive the soft errors in turn reduce the size of the packets they send (i.e., the same as for hard errors), but can soon resume sending larger packets if the soft errors subside.

An OAL source sends PTB soft error messages by setting the ICMPv4 header "unused" field or ICMPv6 header Code field to the value 1 if a original IP packet was deemed lost (e.g., due to reassembly timeout) or to the value 2 otherwise. The OAL source sets the PTB destination address to the original IP packet source, and sets the source address to one of its OMNI interface unicast/anycast addresses that is routable from the perspective of the original source. The OAL source then sets the MTU field to a value smaller than the original packet size but no smaller than 576 for ICMPv4 or 1280 for ICMPv6, writes the leading portion of the original IP packet into the "packet in error" field, and returns the PTB soft error to the original source. When the original source receives the PTB soft error, it temporarily reduces the size of the packets it sends the same as for hard errors but may seek to increase future packet sizes dynamically while no further soft errors are arriving. (If the original source does not recognize the soft error code, it regards the PTB the same as a hard error but should heed the retransmission advice given in [RFC8201] suggesting retransmission based on normal packetization layer retransmission timers.)

An OAL destination may experience reassembly cache congestion, and can return uNA messages to the OAL source that originated the fragments (subject to rate limiting) to advertise reduced hard/soft Reassembly Limits and/or to report individual reassembly failures. The OAL destination creates a uNA message with an OMNI option containing an authentication message sub-option (if the OAL source is on an open Internetwork) followed optionally by at most one hard and one soft Reassembly Limit sub-options with reduced hard/soft values, and with one of them optionally including the leading portion an OAL first fragment containing the header of an original IP packet whose source must be notified (see: Section 12). The OAL destination encapsulates the leading portion of the OAL first fragment (beginning with the OAL header) in the "OAL First Fragment" field of sub-option, signs the message if an authentication sub-option is included, performs OAL encapsulation (with the its own address as the OAL source and the source address of the message that prompted the uNA as the OAL destination) and sends the message to the OAL source.

When the OAL source receives the uNA message, it records the new hard/soft Reassembly Limit values for this OAL destination if the OMNI option includes Reassembly Limit sub-options. If a hard or soft Reassembly Limit sub-option includes an OAL First Fragment, the OAL source next sends a corresponding network layer PTB hard or soft error to the original source to recommend a smaller size. For hard errors, the OAL source sets the PTB Code field to 0. For soft errors, the OAL source sets the PTB Code field to 1 if the L flag in the Reassembly Limit sub-option is 1; otherwise, the OAL source sets the Code field to 2. The OAL source crafts the PTB by extracting the leading portion of the original IP packet from the OAL First Fragment field (i.e., not including the OAL header) and writes it in the "packet in error" field of a PTB with destination set to the original IP packet source and source set to one of its OMNI interface unicast/anycast addresses that is routable from the perspective of the original source. For future transmissions, if the original IP packet is larger than the hard Reassembly Limit for this OAL destination the OAL source drops the packet and returns a PTB hard error with MTU set to the hard Reassembly Limit. If the packet is no larger than the current hard Reassembly Limit but larger than the current soft limit, the OAL source can also return a PTB soft error (subject to rate limiting) with Code set to 2 and MTU set to the current soft limit while still forwarding the packet to the OMNI destination.

Original sources that receive PTB soft errors can dynamically tune the size of the original IP packets they to send to produce the best possible throughput and latency, with the understanding that these parameters may change over time due to factors such as congestion, mobility, network path changes, etc. The receipt or absence of soft errors should be seen as hints of when increasing or decreasing packet sizes may be beneficial. The OMNI interface supports continuous transmission and reception of packets of various sizes in the face of dynamically changing network conditions. Moreover, since PTB soft errors do not indicate a hard limit, original sources that receive soft errors can begin sending larger packets without waiting for the recommended 10 minutes specified for PTB hard errors [RFC1191][RFC8201]. The OMNI interface therefore provides an adaptive service that accommodates MTU diversity especially well-suited for dynamic multilink environments.

6.9. OAL Requirements

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

o OAL sources MUST NOT use the OAL to forward original IP packets larger than the OMNI interface MTU or the OAL destination hard

Reassembly Limit.(i.e., whether as atomic fragments or multiple fragments).

- OAL sources MUST forward original IP packets smaller than the minimum MPS minus the trailer 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 sources MUST NOT produce fragments that include any extension headers other than a single Fragment Header.

- OAL intermediate nodes SHOULD and OAL destinations MUST unconditionally drop any OAL fragments with offset and length that would cause the reassembled packet to exceed the OMNI interface MRU and/or OAL destination hard Reassembly Limit.

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

- OAL intermediate nodes SHOULD and OAL destinations MUST unconditionally drop OAL fragments that include any extension headers other than a single Fragment Header.

- OAL destinations MUST drop any new OAL fragments with Offset and Payload 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 byte original IP packets would require at most 4 OAL fragments, with each non-final fragment containing 400 payload bytes and the final fragment containing 302 payload bytes (i.e., the final 300 bytes of the original IP packet plus the 2 octet trailer). Likewise, maximum-length 9180 byte original IP packets would require at most 23 fragments. For all packet sizes, the likelihood of successful reassembly may improve when the OMNI interface sends all fragments of the same fragmented OAL packet consecutively over the same underlying interface pair instead of spread across multiple underlying 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, the OAL source could impose an inter-fragment delay while the OAL destination is reporting reassembly congestion (see: Section 6.8) and decrease the delay when reassembly congestion subsides.

6.10. OAL Fragmentation Security Implications

As discussed in Section 3.7 of [RFC8900], there are four basic threats concerning IPv6 fragmentation; each of which is addressed by effective mitigations as follows:

1. Overlapping fragment attacks - reassembly of overlapping fragments is forbidden by [RFC8200]; therefore, this threat does not apply to the OAL.

2. Resource exhaustion attacks - this threat is mitigated by providing a sufficiently large OAL reassembly cache and instituting "fast discard" of incomplete reassemblies that may be part of a buffer exhaustion attack. The reassembly cache should be sufficiently large so that a sustained attack does not cause excessive loss of good reassemblies but not so large that (timer-based) data structure management becomes computationally expensive. The cache should also be indexed based on the arrival underlying interface such that congestion experienced over a first underlying interface does not cause discard of incomplete reassemblies for uncongested underlying interfaces.

3. Attacks based on predictable fragment identification values - 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 bytes, 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 bytes 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.

6.11. OAL Super-Packets

By default, the OAL source includes a 40-byte IPv6 encapsulation header for each original IP packet during OAL encapsulation. The OAL source also calculates and appends a 2 octet trailing checksum then performs fragmentation such that a copy of the 40-byte IPv6 header plus an 8-byte IPv6 Fragment Header is included in each OAL fragment (when 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 and trailing checksum. Within the OAL super-packet, the IP header of the first original IP packet (iHa) followed by its data (iDa) is concatenated immediately following the OAL header, then the IP header of the next original packet (iHb) followed by its data (iDb) is concatenated immediately following the first original packet, etc. with the trailing checksum included last. The OAL super-packet format is transposed from [I-D.ietf-intarea-tunnels] and shown in Figure 10:

```
                <------- Original IP packets ------->
                +-----+-----+
                | iHa | iDa |
                +-----+-----+
                      |
                      |     +-----+-----+
                      |     | iHb | iDb |
                      |     +-----+-----+
                      |           |
                      |           |     +-----+-----+
                      |           |     | iHc | iDc |
                      |           |     +-----+-----+
                      |           |           |
                      v           v           v
     +----------+-----+-----+-----+-----+-----+-----+----+
     |  OAL Hdr | iHa | iDa | iHb | iDb | iHc | iDc |Csum|
     +----------+-----+-----+-----+-----+-----+-----+----+
     <--- OAL "Super-Packet" with single OAL Hdr/Csum --->
```

Figure 10: OAL Super-Packet Format

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

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 *NET encapsulations and finally engages the native frame format of the underlying interface. For example, for Ethernet-compatible interfaces the frame format is specified in [RFC2464], for aeronautical radio interfaces the frame format is specified in standards such as ICAO Doc 9776 (VDL Mode 2 Technical Manual), for various forms of tunnels the frame format is found in the appropriate tunneling specification, etc.

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

8. Link-Local Addresses (LLAs)

OMNI interfaces assign IPv6 Link-Local Addresses (LLAs) through pre-service administrative actions. Clients assign "MNP-LLAs" with interface identifiers that embed the MNP, while Proxy/Servers assign "ADM-LLAs" that include an administrative ID guaranteed to be unique on the link. LLAs are configured as follows:

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

o IPv4-compatible MNP-LLAs are constructed as fe80::ffff:[IPv4], i.e., the interface identifier consists of 16 '0' bits, followed by 16 '1' bits, followed by a 32bit IPv4 address/prefix. The prefix length for the LLA is determined by adding 96 to the MNP prefix length. For example, the IPv4-Compatible MNP-LLA for 192.0.2.0/24 is fe80::ffff:192.0.2.0/120, also written as fe80::ffff:c000:0200/120. (The master MNP-LLA for each "/N" prefix sets the final 128-N bits to 0, but all MNP-LLAs that match the prefix are accepted.)

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

Since the prefix 0000::/8 is "Reserved by the IETF" [RFC4291], no MNPs can be allocated from that block ensuring that there is no possibility for overlap between the different MNP- and ADM-LLA constructs discussed above.

Since MNP-LLAs are based on the distribution of administratively assured unique MNPs, and since ADM-LLAs are guaranteed unique through administrative assignment, OMNI interfaces set the autoconfiguration variable DupAddrDetectTransmits to 0 [RFC4862].

Note: If future protocol extensions relax the 64-bit boundary in IPv6 addressing, the additional prefix bits of an MNP could be encoded in bits 16 through 63 of the MNP-LLA. (The most-significant 64 bits would therefore still be in bits 64-127, and the remaining bits would appear in bits 16 through 48.) However, the analysis provided in [RFC7421] suggests that the 64-bit boundary will remain in the IPv6 architecture for the foreseeable future.

Note: Even though this document honors the 64-bit boundary in IPv6 addressing, it specifies prefix lengths longer than /64 for routing purposes. This effectively extends IPv6 routing determination into the interface identifier portion of the IPv6 address, but it does not redefine the 64-bit boundary. Modern routing protocol implementations honor IPv6 prefixes of all lengths, up to and including /128.

9. Unique-Local Addresses (ULAs)

OMNI domains use IPv6 Unique-Local Addresses (ULAs) as the source and destination addresses in OAL packet IPv6 encapsulation headers. ULAs are only routable within the scope of a an OMNI domain, and are derived from the IPv6 Unique Local Address prefix fc00::/7 followed by the L bit set to 1 (i.e., as fd00::/8) followed by a 40-bit pseudo-random Global ID to produce the prefix [ULA]::/48, which is

then followed by a 16-bit Subnet ID then finally followed by a 64 bit Interface ID as specified in Section 3 of [RFC4193]. All nodes in the same OMNI domain configure the same 40-bit Global ID as the OMNI domain identifier. The statistic uniqueness of the 40-bit pseudo-random Global ID allows different OMNI domains to be joined together in the future without requiring renumbering.

Each OMNI link instance is identified by a 16-bit Subnet ID value between 0x0000 and 0xfeff in bits 48-63 of [ULA]::/48. The Subnet ID values 0xff00 through 0xfffe are reserved for future use, while 0xffff denotes the presence of a Temporary ULA (see below). For example, OMNI ULAs associated with instance 0 are configured from the prefix [ULA]:0000::/64, instance 1 from [ULA]:0001::/64, instance 2 from [ULA]:0002::/64, etc. ULAs and their associated prefix lengths are configured in correspondence with LLAs through stateless prefix translation where "MNP-ULAs" are assigned in correspondence to MNP-LLAs and "ADM-ULAs" are assigned in correspondence to ADM-LLAs. For example, for OMNI link instance [ULA]:1010::/64:

o the MNP-ULA corresponding to the MNP-LLA fe80::2001:db8:1:2 with a 56-bit MNP length is derived by copying the lower 64 bits of the LLA into the lower 64 bits of the ULA as [ULA]:1010:2001:db8:1:2/120 (where, the ULA prefix length becomes 64 plus the IPv6 MNP length).

o the MNP-ULA corresponding to fe80::ffff:192.0.2.0 with a 28-bit MNP length is derived by simply writing the LLA interface ID into the lower 64 bits as [ULA]:1010:0:ffff:192.0.2.0/124 (where, the ULA prefix length is 64 plus 32 plus the IPv4 MNP length).

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

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

o etc.

The ULA presents an IPv6 address format that is routable within the OMNI routing system and can be used to convey link-scoped IPv6 ND messages across multiple hops using IPv6 encapsulation [RFC2473]. The OMNI link extends across one or more underling Internetworks to include all Proxy/Servers. 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).

Temporary ULAs are constructed per [RFC8981] based on the prefix [ULA]:ffff::/64 and used by Clients when they have no other

addresses. Temporary ULAs can be used for Client-to-Client communications outside the context of any supporting OMNI link infrastructure, and can also be used as an initial address while the Client is in the process of procuring an MNP. Temporary ULAs are not routable within the OMNI routing system, and are therefore useful only for OMNI link "edge" communications. Temporary ULAs employ optimistic DAD principles [RFC4429] since they are probabilistically unique.

Each OMNI link may be subdivided into SRT segments that often correspond to different administrative domains or physical partitions. 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).

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 domain are represented by non-MNP GUA prefixes that are not derived from the MSP, but are treated in all other ways the same as for MNPs.

For IPv6, GUA MSPs are assigned by IANA [IPV6-GUA] and/or an associated regional assigned numbers authority such that the OMNI domain can be interconnected to the global IPv6 Internet without causing inconsistencies in the routing system. An OMNI domain could instead use ULAs with the 'L' bit set to 0 (i.e., from the prefix fc00::/8)[RFC4193], however this would require IPv6 NAT if the domain were ever connected to the global IPv6 Internet.

For IPv4, GUA MSP are assigned by IANA [IPV4-GUA] and/or an associated regional assigned numbers authority such that the OMNI domain can be interconnected to the global IPv4 Internet without causing routing inconsistencies. An OMNI domain could instead use private IPv4 prefixes (e.g., 10.0.0.0/8, etc.) [RFC3330], however this would require IPv4 NAT if the domain were ever connected to the global IPv4 Internet. OMNI interfaces advertise IPv4 MSPs into IPv6 routing systems as IPv4-mapped IPv6 prefixes [RFC4291] (e.g., the IPv6 prefix for the IPv4 MSP 192.0.2.0/24 is ::ffff:192.0.2.0/120) .

OMNI interfaces assign the IPv4 anycast address 192.88.99.1, and IPv4 routers that configure OMNI interfaces advertise the prefix 192.88.99.0/24 into the routing system of other networks (see: IANA Considerations). Specific applications for OMNI IPv6 and IPv4 anycast addresses are discussed throughout the document.

OMNI interfaces also configure global IPv6 anycast addresses based on the prefix 2002:c058:6301::/48, which is the IPv6 derivation of the OMNI IPv4 anycast address (see above). OMNI IPv6 anycast addresses are formed as:

2002:c058:6301:MNP[64]:Preflen[8]:Link_ID[8]

where MNP[64] encodes an MSP up to 64 bits in length, Preflen[8] encodes the length of the prefix and Link_ID[8] encodes a value between 0-254 that identifies a specific OMNI link within an OMNI domain (the Link_ID value 255 is an OMNI link "anycast" value configured by all OMNI interfaces within the same domain). For example, the OMNI IPv6 anycast address for MSP 2001:db8::/32 is 2002:c058:6301:2001:db8:0:0:32[Link_ID], the OMNI IPv6 anycast address for MSP 192.0.2.0/24 is 2002:c058:6301:0000:ffff:c000:0200:24[Link_ID], etc.).

OMNI interfaces assign OMNI IPv6 anycast addresses, and IPv6 routers that configure OMNI interfaces advertise the corresponding prefixes into the routing system of other networks. An OMNI IPv6 anycast prefix is formed the same as for any IPv6 prefix; for example, the prefix 2002:c058:6301:2001:db8::/80 matches all OMNI IPv6 anycast addresses covered by the prefix. By advertising OMNI IPv6 anycast prefixes in this way, OMNI Clients can locate and associate with the OMNI domain and/or a specific link within the OMNI domain that services the MSP of interest.

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 an IPv6 temporary ULA or (H)HIT as an IPv6 source/destination address for sustained communications in Vehicle-to-Vehicle (V2V) and (multihop) Vehicle-to-Infrastructure (V2I) scenarios. The 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.) 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 neighbor cache for tracking per-neighbor state and use the link-local address format specified in Section 8. IPv6 Neighbor Discovery (ND) [RFC4861] messages sent over OMNI interfaces without encapsulation observe the native underlying interface Source/Target Link-Layer Address Option (S/TLLAO) format (e.g., for Ethernet the S/TLLAO is specified in [RFC2464]). IPv6 ND messages sent over OMNI interfaces using encapsulation do not include S/TLLAOs, but instead include a new option type that encodes encapsulation addresses, interface attributes and other OMNI link information. Hence, this document does not define an S/TLLAO format but instead defines a new option type termed the "OMNI option" designed for these purposes. (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 entire length of the message (beginning with a pseudo-header of the IPv6 header) but does not calculate/include the IPv6 ND message checksum itself. If the OMNI interface forwards the message to a next hop over the secured spanning tree path, it need not include either an authentication signature or checksum since lower layers already ensure authentication and integrity. In all other cases, the OMNI interface calculates the standard IPv6 ND message checksum 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 many wireless data link types (e.g. satellite-based, cellular, terrestrial, air-to-air directional, etc.) with diverse performance, cost and availability properties. The OMNI interface would therefore appear to have multiple L2 connections, and may include information for multiple underlying interfaces in a single IPv6 ND message exchange. OMNI interfaces 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

The first OMNI option appearing in an IPv6 ND message is formatted as shown in Figure 11:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |     Length    |    Preflen    | S/T-omIndex  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Acknowledgment Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|A|R|O|P|     |                                               |
|Y|C|S|P|N| Res |                    Window                     |
|N|K|T|T|G|     |                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                          Sub-Options                          ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: OMNI Option Format

In this format:

o Type is set to TBD3.

o Length is set to the number of 8 octet blocks in the option. The value 0 is invalid, while the values 1 through 255 (i.e., 8 through 2040 octets, respectively) indicate the total length of the OMNI option.

o Preflen is an 8 bit field that determines the length of prefix associated with an LLA. Values 0 through 128 specify a valid prefix length (all other values are invalid). For IPv6 ND messages sent from a Client to the MS, Preflen applies to the IPv6 source LLA and provides the length that the Client is requesting or asserting to the MS. For IPv6 ND messages sent from the MS to the Client, Preflen applies to the IPv6 destination LLA 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.

o S/T-omIndex is an 8 bit field that includes an omIndex value for the source or target underlying interface for this IPv6 ND message. Client OMNI interfaces MUST number each distinct underlying interface with an omIndex value between '1' and '255' that represents a Client-specific 8-bit mapping for the actual ifIndex value assigned by network management [RFC2863], then set S/T-omIndex to either a specific omIndex value or '0' to denote "unspecified". Proxy/Server OMNI interfaces use the omIndex value '0' to denote an INET underlying interface and/or to inform a peer Proxy/Server that a Client has departed.

o The remaining header fields before "Sub-Options" are modeled from the Transmission Control Protocol (TCP) header specified in Section 3.1 of [RFC0793] and include a 32 bit Sequence Number followed by a 32 bit Acknowledgement Number followed by 8 flags bits followed by a 24-bit Window. 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.

o 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. If multiple OMNI option instances appear in the same IPv6 ND message, only the first option includes the OMNI header fields before the Sub-Options while all others are coded as follows:

```
  0                   1                   2
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
 |     Type      |     Length    | Sub-Options ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 type-length-value (TLV) format encoded as follows:

```
  0                   1                   2
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
 | Sub-Type|      Sub-length     | Sub-Option Data ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 12: Sub-Option Format

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

Sub-Option Name	Sub-Type
Pad1	0
PadN	1
Multilink Fwding Parameters	2
Interface Attributes	3
Traffic Selector	4
Geo Coordinates	5
DHCPv6 Message	6
HIP Message	7
PIM-SM Message	8
Reassembly Limit	9
Fragmentation Report	10
Node Identification	11
ICMPv6 Error	12
Sub-Type Extension	30

Figure 13

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

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

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

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 header and any preceding sub-options for this OMNI option. 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 for that instance and ignores the final sub-option. 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 12.2.8), it MUST appear as the first sub-option of the first OMNI option which must appear immediately following the IPv6 ND message header. If the IPv6 ND message includes additional authentication sub-options, only the first sub-option is processed and all others are ignored.

When a Client OMNI interface prepares an RS or secured NS message, it includes a Mutilink Forwarding Parameters sub-option specific to the underlying interface that will transmit the RS/NS (see: Section 12.2.3) immediately following the authentication sub-option if present; otherwise as the first sub-option of the first OMNI option which must appear immediately following the IPv6 ND message header.

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

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

Figure 14: Pad1

- o Sub-Type is set to 0. If multiple instances appear in OMNI options of the same message all are processed.
- o Sub-Type is followed by 3 'x' bits, set to any value on transmission (typically all-zeros) and ignored on 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

```
 0                   1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| S-Type=1|    Sub-length=N     | N padding octets ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 15: PadN

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

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

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

When a proxy forwards an IPv6 ND message with OMNI options, it can employ PadN to cancel 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 but need not be re-calculated if authentication and integrity assurance will be applied by lower layers on the path to the next hop. See: Appendix B for a discussion of IPv6 ND message authentication and integrity.

12.2.3. Interface Attributes

The Interface Attributes sub-option provides forwarding information for the multilink conceptual sending algorithm discussed in Section 14. The forwarding information is used for selecting among potentially multiple candidate underlying interfaces that can be used to forward carrier packets to the neighbor based on factors such as traffic selectors and link quality. Interface Attributes further includes 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.

Hub Proxy/Servers include Interface Attributes for all of a target Client's underlying interfaces in NA Address Resolution messages. Proxy/Servers also include Interface Attributes for all of a target Client's underlying interfaces in uNA messages used to publish Client information changes (see: [I-D.templin-6man-aero] for more information). When the node that sent the NS message receives the NA, it can use all of the included Interface Attributes and/or Traffic Selectors to formulate a map of the prospective target node as well as to seed the information to be populated in a Multilink Forwarding Parameters sub-option.

Interface Attributes must be honored by all implementations in the format shown below:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| S-Type=2|    Sub-length=N     |    omIndex    |     omType    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Provider ID  | Link  | Resvd | FMT |   SRT   |     LHS (0)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     LHS (1-3)                 |               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               ~
~                                                               ~
~                   Link Layer Address (L2ADDR)                 ~
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Interface Attributes

- o Sub-Type is set to 2. NA messages used for Address Resolution and uNA messages include Interface Attributes for all of the target Client's underlying interfaces. If the IPv6 ND message includes multiple Interface Attributes instances with the same omIndex value (whether in the same OMNI option or additional OMNI options), only the first instance is processed and all others are ignored.

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

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

  * omIndex is a 1-octet value corresponding to a specific underlying interface the same as specified above for the OMNI option S/T-omIndex field. The OMNI options of a single message may include multiple Interface Attributes sub-options, with each distinct omIndex value pertaining to a different underlying interface.

  * omType is set to an 8-bit integer value corresponding to the underlying interface identified by omIndex. The value represents an OMNI interface-specific 8-bit mapping for the actual IANA ifType value registered in the 'IANAifType-MIB' registry [http://www.iana.org].

  * Provider ID is set to an OMNI interface-specific 8-bit ID value for the network service provider associated with this omIndex.

* Link encodes a 4-bit link metric. The value '0' means the link is DOWN, and the remaining values mean the link is UP with metric ranging from '1' ("lowest") to '15' ("highest").

* Resvd is 4-bit field reserved for future use, set to 0 on transmit and ignored on receipt.

* FMT - a 3-bit "Forward/Mode/Type" code interpreted as follows:

  + When the most significant bit (i.e., "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 underlying 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 Proxy/Server; otherwise the Client is eligible for direct forwarding over the open INET where it may be located behind one or more NATs.

  + The next most significant bit (i.e., "FMT-Mode") is interpreted in conjunction with the FMT-Forward bit, as discussed above.

  + The least significant bit (i.e., "FMT-Type") determines the IP address version encoded in L2ADDR. If FMT-Type is clear, L2ADDR includes a 4-octet IPv4 address. If FMT-Type is set, L2ADDR includes a 16-octet IPv6 address.

* SRT - a 5-bit Segment Routing Topology prefix length value that (when added to 96) determines the prefix length to apply to the ULA formed from concatenating [ULA*]::/96 with the 32 bit LHS MSID value that follows. For example, the value 16 corresponds to the prefix length 112.

* LHS - the 32 bit MSID of the LHS Proxy/Server on the path to the target. When SRT and LHS are both set to 0, the LHS Proxy/Server is considered unspecified in this IPv6 ND message. SRT and LHS together provide guidance for the OMNI interface forwarding algorithm. Specifically, if SRT/LHS is located in the local OMNI link segment then the target Client can be reached either through its dependent LHS Proxy/Server or

directly following NAT traversal conversion. Otherwise, the target Client is located on a different SRT segment and must be reached via the spanning tree. See [I-D.templin-6man-aero] for further discussion.

* Link Layer Address (L2ADDR) - identifies the link-layer address (i.e., the encapsulation address) of the source/target according to FMT. The first 2 octets encodes a UDP port number, and an IP address appears in the next 4 octets for IPv4 or 16 octets for IPv6. The UDP port number and IP address are recorded in network byte order, and in ones-compliment "obfuscated" form per [RFC4380].

12.2.4. Multilink Forwarding Parameters

OMNI nodes include the Multilink Forwarding Parameters sub-option in NS/NA messages used to coordinate with multilink route optimization targets, or in RS/RA messages used to coordinate with (remote) Proxy/Servers. If a solicitation message includes the sub-option, the solicited advertisement response must also include the sub-option. The OMNI node MUST include the sub-option in the first OMNI option immediately following the HIP message sub-option and/or a single Pad1/PadN if present. Otherwise, the OMNI node MUST include the sub-option immediately following the OMNI header.

The Multilink Forwarding Parameters sub-option includes the necessary state for establishing Multilink Forwarding Vectors (MFVs) in the Multilink Forwarding Information Bases (MFIBs) of the OAL source, destination and all intermediate nodes in the path. The manner for populating MFIB/MFV information is specified in detail in [I-D.templin-6man-aero].

The Multilink Forwarding Parameters sub-option is formatted as shown in Figure 17:

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | S-Type=3|    Sub-length=N     |FHS Cli omIndex|     omType    |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |  Provider ID  | Link  | Resvd | FMT |   SRT   |               ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               ~
       ~                   FHS Client UDP Port/INADDR                  ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       ~                  FHS Proxy/Server MSID/INADDR                 ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       ~                     FHS Bridge MSID/INADDR                    ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |LHS Cli omIndex|     omType    |  Provider ID  | Link  | Resvd |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | FMT |   SRT   |                                               ~
       +-+-+-+-+-+-+-+-+                                               ~
       ~                   LHS Client UDP Port/INADDR                  ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       ~                  LHS Proxy/Server MSID/INADDR                 ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       ~                     LHS Bridge MSID/INADDR                    ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       ~           Tunnel Window Synchronization Parameters            ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       ~                                                               ~
       ~        Multilink Forwarding Vector Index (MFVI) List          ~
       ~               (5 consecutive 4-octet MFVIs)                   ~
       ~                                                               ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |  A  |  B  |Job|
       +-+-+-+-+-+-+-+-+
```

Figure 17: Multilink Forwarding Parameters

- Sub-Type is set to 3. 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 is set to N that encodes the number of Sub-Option Data octets that follow.

- Sub-Option Data contains Multilink Forwarding Parameters as follows:

  * FHS Client omIndex, omType, Provider ID and Link/Reserved are fields (at offset 0 from the beginning of the Sub-Option Data)

that include link parameters for the FHS Client underlying interface. (This is the same information that would appear in an Interface Attributes sub-option.)

* (FHS) FMT/SRT is a 1-octet field that applies to the FHS information. The SRT prefix length information applies to all FHS elements since all are by definition in the same SRT segment. The FMT-Forward/Mode bits determine the characteristics of the FHS Proxy/Server relationship for this specific FHS Client underlying interface (i.e., the same as described in Section 12.2.3), and the FMT-Type bits determine the IP address version for all INADDR fields relative to this SRT segment. Unlike the case for Interface Attributes, all INADDR fields are always 16 bits in length regardless of the IP protocol version (for IPv4, INADDR is encoded as an IPv4-mapped IPv6 address [RFC4291]). The IP address (as well as UDP port number when present) is recoded in network byte order, and in ones-compliment "obfuscated" form the same as described in Section 12.2.3.

* FHS Client UDP Port/INADDR includes the *NET encapsulation 2-octet UDP port number followed by the 16-octet INADDR observed by the FHS Proxy/Server when it processes an IPv6 ND solicitation message sent by the FHS Client containing this option. When an FHS Client RS message includes a non-zero UDP Port and INADDR, the FHS Proxy/Server that receives the RS should compare the UDP/INADDR values with the actual *NET encapsulation addresses; if the addresses differ the presence of a NAT is indicated.

* FHS Proxy/Server MSID/INADDR includes a 4-octet FHS Proxy/ Server MSID followed by a 16 octet INADDR the same as above. INADDR identifies an open INET interface not located behind NATs, therefore no UDP port number is included since port number 8060 is used when the *NET encapsulation includes a UDP header.

* FHS Bridge MSID/INADDR encodes a 4 octet MSID followed by a 16-octet INADDR exactly as for the FHS Proxy/Server MSID/ INADDR.

* LHS Client omIndex, omType, Provider ID, Link/Reserved, FMT/ SRT, Client UDP/INADDR, Proxy/Server MSID/INADDR and Bridge MSID/INADDR are coded exactly the same as for their FHS counterparts above except that they provide information for LHS elements.

* 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). End systems can therefore use the OMNI header parameters for end-to-end window synchronization while tunnel endpoints use the tunnel parameters for simultaneous middlebox window synchronization in a single NS/NA message exchange. The Tunnel Window Synchronization Parameters block offset is 33 octets before the end of the Sub-Option Data.

* Multilink Forwarding Vector Index (MFVI) List is a list of at most 5 consecutive 4-octet MFVIs. The FHS/LHS source and each intermediate node on the path to the destination processes the list according to the A, B and Job codes (see below).

* A is a 3-bit count of the number of "A" MVFI List entries (valid values are 0-5).

* B is a 3-bit count of the number of "B" MVFI List entries (valid values are 0-5).

* Job is a 2-bit code that determines the manner in which each node in the path processes the MVFI List as follows:

  + 00 - "Initialize; Build B" - the FHS source sets this code in a solicitation used to initialize MFV 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 MFV. Each node that processes the message then assigns a unique 4-octet "B" MFVI to the MVF 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 MFVI 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).

  + 01 - "Follow B; Build A" - the LHS source sets this code in a solicited advertisement response to a solicitation with code "00" (any other messages that include this code MUST be dropped). The LHS source first copies the MFVI 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 MFVI List entry B to locate the corresponding MFV. Each node that processes the message then assigns a unique 4-octet "A" MFVI to the MVF 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 MFVI 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).

+ 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 solicitations/advertisements sent to the same LHS destination. The FHS source copies the MVFI 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" MFVI found at list entry B to locate the corresponding MFV. Each node that processes the message then writes the MVF's "B" MFVI into list entry B, then decrements A and increments B. When the message arrives at the LHS destination, B will contain the number of MFVI List "B" entries populated in the forward direction.

+ 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 solicitations/advertisements sent to the same FHS destination. The LHS source copies the MVFI 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" MFVI List entry found at list entry B to locate the corresponding MFV. Each node that processes the message then writes the MFV's "A" MFVI into list entry B, then increments A and decrements B. When the message arrives at the FHS destination, A will contain the number of MFVI List "A" entries populated in the reverse direction.

A, B and Job 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 MVF. When a Job code specifies "Build", each node that processes the message assigns a new MFVI. When a Job code specifies

"Follow", each node that processes the message uses an A/B MFVI List entry to locate an MFV (if the MFV 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 MFVI List to determine the number of previous hops, the number of remaining hops, and the A/B MFVIs associated with the previous/remaining hops. However, no intermediate nodes will discover inappropriate A/B MFVIs for their location in the multihop forwarding chain. See: [I-D.templin-6man-aero] for further discussion on A/B MFVI processing.

12.2.5. Traffic Selector

When used in conjunction with Interface Attributes and/or Multilink Forwarding Parameters information, the Traffic Selector sub-option provides forwarding information for the multilink conceptual sending algorithm discussed in Section 14.

Clients include Traffic Selector sub-options specific to the omIndexes of underlying interfaces serviced by the same FHS/Hub Proxy/Servers. Prospective peer Clients that receive the Traffic Selectors in NA messages can then use them to drive the multilink forwarding algorithm.

Proxy/Servers include Traffic Selectors for all of a target Client's underlying interfaces in NA Address Resolution messages. Proxy/Servers also include Traffic Selectors for all of a target Client's underlying interfaces in uNA messages used to publish Client information changes. See: [I-D.templin-6man-aero] for more information.

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

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | S-Type=4|    Sub-length=N     |    omIndex     |   TS Format   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       ~                                                               ~
       ~                RFC 6088 Format Traffic Selector               ~
       ~                                                               ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 18: Traffic Selector

o Sub-Type is set to 4. Each IPv6 ND message may contain zero or more Traffic Selectors for each omIndex; when multiple Traffic Selectors for the same omIndex appear, all are processed and the cumulative information from all is retained.

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

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

  * omIndex is a 1-octet value corresponding to a specific underlying interface the same as specified above for the OMNI option S/T-omIndex field. The OMNI options of a single message may include multiple Traffic Selector sub-options, with each distinct omIndex value pertaining to a different underlying interface.

  * TS Format is a 1-octet field that encodes a Traffic Selector version per [RFC6088] when T is 1. If TS Format encodes the value 1 or 2, the Traffic Selector includes IPv4 or IPv6 information, respectively. If TS Format encodes the value 0, the Traffic Selector field is omitted.

  * When TS Format is non-zero, 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 omIndex requires Traffic Selectors for multiple IP protocol versions, or if a Traffic Selector block would exceed the space available in a single Interface Attributes sub-option, the remaining information is coded in additional Traffic Selector sub-options that all encode the same omIndex.

12.2.6. Geo Coordinates

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| S-Type=5|    Sub-length=N     |    Geo Type   |Geo Coordinates
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ...
```

Figure 19: Geo Coordinates Sub-option

- Sub-Type is set to 5. 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.

12.2.7. 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.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| S-Type=6|    Sub-length=N     |    msg-type   |  id (octet 0) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   transaction-id (octets 1-2) |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
.                        DHCPv6 options                         .
.                 (variable number and length)                  .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 20: DHCPv6 Message Sub-option

- o Sub-Type is set to 6. If multiple instances appear in OMNI options of the same message the first is processed and all others are ignored.

- o Sub-Length is set to N 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.

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

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

12.2.8. Host Identity Protocol (HIP) Message

The Host Identity Protocol (HIP) Message sub-option should be included in OMNI options to provide 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 insert HIP authentication signatures before forwarding them to the target Client.

OMNI interfaces MUST include the HIP message 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 over unsecured paths without a HIP message (or other

authentication 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 underlying interfaces, i.e., where authentication and integrity is not already assured by lower layers. OMNI interfaces that process 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 path MPS. 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 bytes 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:

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | S-Type=7|    Sub-length=N     |0| Packet Type |Version| RES.|1|
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |           Reserved            |           Controls            |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                Sender's Host Identity Tag (HIT)               |
       |                                                               |
       |                                                               |
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |               Receiver's Host Identity Tag (HIT)              |
       |                                                               |
       |                                                               |
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                               |
       /                        HIP Parameters                         /
       /                                                               /
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: HIP 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, 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, other types of IPv6 addresses (e.g., a Client's MNP-LLA, a Proxy/Server's ADM-LLA,

etc.) can be used instead of HITs in the authentication signature as long as the address can be uniquely associated with the Sender/ Receiver.

12.2.9. 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 22:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| S-Type=8|    Sub-length=N     |PIM Ver| Type  |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                         PIM-SM Message                        /
/                                                               /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 22: PIM-SM Message Option Format

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

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

- o 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.)

12.2.10. Reassembly Limit

The Reassembly Limit sub-option may be included in the OMNI options of IPv6 ND messages. The message consists of a 15-bit Reassembly Limit value, followed by a flag bit (H) optionally followed by an (N-2)-octet leading portion of an OAL First Fragment that triggered the message.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| S-Type=9|    Sub-length=N     |      Reassembly Limit       |H|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          OAL First Fragment (As much of invoking packet       |
+         as possible without causing the IPv6 ND message       +
|                to exceed the minimum IPv6 MTU)                |
+                                                               +
```

Figure 23: Reassembly Limit

- o Sub-Type is set to 9. If multiple instances appear in OMNI options of the same message the first occurring "hard" and "soft" Reassembly Limit values are accepted, and any additional Reassembly Limit values are ignored.

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

- o A 15-bit Reassembly Limit follows, and includes a value between 1500 and 9180. If any other value is included, the sub-option is ignored. The value indicates the hard or soft limit for original IP packets that the source of the message is currently willing to reassemble; the source may increase or decrease the hard or soft limit at any time through the transmission of new IPv6 ND messages. Until the first IPv6 ND message with a Reassembly Limit sub-option arrives, OMNI nodes assume initial default hard/soft limits of 9180 (I.e., the OMNI interface MRU). After IPv6 ND messages with Reassembly Limit sub-options arrive, the OMNI node retains the most recent hard/soft limit values until new IPv6 ND messages with different values arrive.

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

- o If N is greater than 2, the remainder of the Reassembly Limit sub-option encodes the leading portion of an OAL First Fragment that prompted this IPv6 ND message. The first fragment is included beginning with the OAL IPv6 header, and continuing with as much of

the fragment payload as possible without causing the IPv6 ND message to exceed the minimum IPv6 MTU.

12.2.11. Fragmentation Report

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

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |S-Type=10|   Sub-Length = N    | Identification #1 (bits 0 -15)|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Identification #1 (bits 15-31)|    Bitmap #1 (bits 0 - 15)    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Bitmap #1 (bits 16-31)  | Identification #2 (bits 0 -15)|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Identification #2 (bits 15-31)|    Bitmap #2 (bits 0 - 15)    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Bitmap #2 (bits 16-31)  | Identification #3 (bits 0 -15)|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Identification #3 (bits 15-31)|    Bitmap #3 (bits 0 - 15)    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Bitmap #3 (bits 16-31)  |             ...               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+             ...               +
 |                              ...                              |
 +                              ...                              +
```

Figure 24: Fragmentation Report

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

- o Sub-Length is set to N, i.e., the length of the option in octets beginning immediately following the Sub-Length field and extending to the end of the sub-option. If N is not an integral multiple of 8 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 MPS.

- o Identification (i) includes the IPv6 Identification value found in the Fragment Header of a received OAL fragment. (Only those Identification values included represent fragments for which loss was unambiguously observed; any Identification values not included

correspond to fragments that were either received in their entirety or may still be in transit.)

o Bitmap (i) includes an ordinal checklist of fragments, with each bit set to 1 for a fragment received or 0 for a fragment missing. (Each OAL packet may consist of at most 23 fragments, therefore Bitmap (i) bits 0-22 are consulted while bits 23-31 are reserved for future use and ignored.) 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:

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

Figure 25

(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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S-Type=11|    Sub-length=N    |      ID-Type    |               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+              ~
~            Node Identification Value (N-1 octets)              ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 26: Node Identification

o 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.)

o 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.

o ID-Type is a 1 octet field that encodes the type of the Node Identification Value. The following ID-Type values are currently defined:

  * 0 - Universally Unique IDentifier (UUID) [RFC4122]. Indicates that Node Identification Value contains a 16 octet UUID.

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

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

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

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

  * 5 - 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.

o 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 27:

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         DUID-Type (2)         |      EN (high bits == 0)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     EN (low bits = 45282)     |    ID-Type    |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               |
.                    Node Identification Value                  .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 27: 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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S-Type=12|    Sub-length=N     |                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-                                ~
~                    RFC4443 Error Message Body                 ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 28: ICMPv6 Error

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

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

- o RFC4443 Error Message Body is an N-octet field encoding the body of an ICMPv6 Error Message per Section 2.1 of [RFC4443] (ICMPv6 informational messages must not be included and must be ignored if received). OMNI interfaces include as much of the ICMPv6 error message body in the sub-option as possible without causing the IPv6 ND message to exceed the minimum IPv6 MTU.

12.2.14. Sub-Type Extension

Since the Sub-Type field is only 5 bits in length, future specifications of major protocol functions may exhaust the remaining Sub-Type values available for assignment. This document therefore defines Sub-Type 30 as an "extension", meaning that the actual Sub-Option type is determined by examining a 1 octet "Extension-Type"

field immediately following the Sub-Length field. The Sub-Type Extension is formatted as shown in Figure 29:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S-Type=30|     Sub-length=N    | Extension-Type|               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               ~
~                                                               ~
~                      Extension-Type Body                      ~
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 29: Sub-Type Extension

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

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

12.2.14.1. RFC4380 Header Extension Option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S-Type=30|      Sub-length=N   |   Ext-Type=0  |   Header Type |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                      Header Option Value                      ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

o Sub-Type is set to 30.

o 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.

o Extension-Type is set to 0. Each instance encodes exactly one header option per Section 5.1.1 of [RFC4380], with the leading '0' octet omitted and the following octet coded as Header Type. If multiple instances of the same Header Type appear in OMNI options of the same message the first instance is processed and all others are ignored. 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.

o Header Type and Header Option Value are coded exactly as specified in Section 5.1.1 of [RFC4380]; the following types are currently defined:

   * 0 - Origin Indication (IPv4) - value coded per Section 5.1.1 of [RFC4380].

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

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

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

12.2.14.2. RFC6081 Trailer Extension Option

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |S-Type=30|      Sub-length=N   |   Ext-Type=1  |  Trailer Type |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       ~                     Trailer Option Value                      ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

o Sub-Type is set to 30.

o 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.

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

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

   * 0 - Unassigned

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

   * 2 - Unassigned

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

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

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

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

o Trailer Type values 7 through 252 are available for assignment by future specifications, which must also define the format of the Trailer Option Value and its processing rules. Trailer Type values 253 and 254 are reserved for experimentation, as recommended in [RFC3692], and value 255 is Reserved by IANA.

13. Address Mapping - Multicast

The multicast address mapping of the native underlying interface applies. The Client mobile router also serves as an IGMP/MLD Proxy for its EUNs and/or hosted applications per [RFC4605].

The Client uses Multicast Listener Discovery (MLDv2) [RFC3810] to coordinate with Proxy/Servers, and *NET L2 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 EUN applications to external correspondents. Each OMNI interface maintains a neighbor cache the same as for any IPv6 interface, but 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 underlying interfaces based on PBM factors (e.g., traffic attributes, cost, performance, message size, etc.) and one or more target underlying interfaces for the neighbor based on Interface Attributes received in IPv6 ND messages (see: Section 12.2.3). Multilink forwarding may also direct packet replication across multiple underlying 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 underlying interfaces.

When the OMNI interface sends an original IP packet over a selected source underlying interface, it first employs OAL encapsulation and fragmentation as discussed in Section 5, then performs *NET

encapsulation as directed by the appropriate MFV. The OMNI interface also performs *NET 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 underlying 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 EUN routing system. Multiple distinct OMNI links can therefore be used to support fault tolerance, load balancing, reliability, etc.

Applications in EUNs 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.

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 internal networks.

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 LLA, 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 interface with the MS by sending RS messages with OMNI options under the assumption that a Proxy/Server on the *NET will process the message and respond. The RS message is received by an "FHS" Proxy/Server, which may in turn forward a proxyed copy of the RS to the Client's current Hub Proxy/Server. The Client then configures default routes for the OMNI interface based on any RA message responses.

For each underlying interface, the Client sends RS messages with OMNI options to coordinate with FHS Proxy/Servers and a single Hub Proxy/Server identified by MSID values. Example MSID discovery methods are given in [RFC5214] and include data link login parameters, name service lookups, static configuration, a static "hosts" file, etc. When the Client sends an RS message to a new FHS Proxy/Server, it first generates an MFVI then includes an OMNI option with an authentication signature if necessary and a Multilink Forwarding Parameters sub-option for the source underlying interface. The RS message includes All-Routers (link) multicast or a unicast ADM-LLA as the RS destination address, and includes an OMNI IPv6 anycast address or a specific unicast ADM-ULA as the OAL destination address when OAL encapsulation is used.

When an FHS Proxy/Server receives an RS with destination set to its own ADM-LLA or All-Routers multicast, it authenticates the message then assumes the Hub Proxy/Server role and processes the message locally. The Hub Proxy/Server creates a NCE for the Client and caches the information in the Multilink Forwarding Parameters and any Traffic Selector sub-options, then acts as the sole entry point for injecting the Client's MNP into the MSE routing system (i.e., after performing any necessary MNP prefix delegation operations). The Hub Proxy/Server then prepares to return an RA message directly to the Client.

When an FHS Proxy/Server receives an RS with destination set to the ADM-LLA of another Proxy/Server acting as the Hub, the FHS Proxy/Server authenticates and proxies the message. The FHS Client includes a Multilink Forwarding Parameters sub-option in the RS for its underlying interfaces serviced by this FHS Proxy/Server, and the FHS Proxy/Server must write the FHS Client's *NET addresses and its own parameters in the appropriate sub-option fields. The FHS Proxy/Server then re-encapsulates the RS in an OAL header with source set to its own ADM-ULA and destination set to the ADM-ULA of the Hub Proxy/Server then forwards the RS over the SRT secured spanning tree.

When the Hub Proxy/Server receives the RS, it caches any state (including Multilink Forwarding information, Traffic Selectors and window synchronization parameters) and performs any necessary prefix delegation and injection. The Hub Proxy/Server then returns an RA via the secured spanning tree with its own ADM-ULA as the OAL source, the ADM-ULA of the FHS Proxy/Server as the OAL destination, with a Multilink Forwarding Parameters sub-option that includes its own parameters in the appropriate sub-option fields. When the FHS Proxy/Server receives the RA, it re-encapsulates in a new OAL header with source set to its own ADM-ULA and destination set to the MNP-ULA of the Client while including an authentication signature if necessary.

Clients configure OMNI interfaces that observe the properties discussed in the previous section. The OMNI interface and its underlying interfaces are said to be in either the "UP" or "DOWN" state according to administrative actions in conjunction with the interface connectivity status. An OMNI interface transitions to UP or DOWN through administrative action and/or through state transitions of the underlying interfaces. When a first underlying interface transitions to UP, the OMNI interface also transitions to UP. When all underlying interfaces transition to DOWN, the OMNI interface also transitions to DOWN.

When a Client OMNI interface transitions to UP, it sends RS messages to register its MNP and an initial set of underlying interfaces that are also UP. The Client sends additional RS messages to refresh lifetimes and to register/deregister underlying interfaces as they transition to UP or DOWN. The Client's OMNI interface sends initial RS messages over an UP underlying interface with its MNP-LLA as the source (or with the unspecified address (::) as the source if it does not yet have an MNP-LLA) and with destination set to link-scoped All-Routers multicast (ff02::2) [RFC4291]. The OMNI interface includes an OMNI option per Section 12 with a Preflen assertion, Multilink Forwarding Parameters appropriate for the underlying interface, Reassembly Limits, and with any other necessary OMNI sub-options (e.g., an authentication sub-option). The OMNI interface then sets the S/T-omIndex field to identify the underlying interface used to forward the RS message.

The OMNI interface then forwards the RS over the underlying interface using OAL encapsulation and fragmentation if necessary. If the Client uses OAL encapsulation for RS messages sent to an unsynchronized INET interface neighbor, 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 reassemble. The OMNI interface selects an Identification value (see: Section 6.6), sets the OAL source address to the ULA corresponding to the RS source (or a Temporary ULA if the RS source

is the unspecified address (::)) and sets the OAL destination to an OMNI IPv6 anycast or ADM-ULA unicast address then sends the message.

FHS Proxy/Servers reached via the underlying interface receive IPv6 ND messages with OMNI options and create a NCE for the Client if necessary while coordinating with a Hub Proxy/Server as discussed above. When the Hub Proxy/Server processes the RS OMNI information, it first validates the prefix registration information then injects/ withdraws the MNP in the MS as necessary and caches/discards the new Preflen, MNP and Multilink Forwarding Parameters. The Hub Proxy/ Server then returns an RA message with an OMNI option per Section 12.

The Hub Proxy/Server returns each RA to the FHS Proxy/Server for the specific Client underlying interface, and the FHS Proxy/Server returns a proxyed version of the RA to the Client via the same underlying interface over which the RS was received. Each RA message includes the Client's MNP-LLA as the destination, the ADM-LLA of Hub Proxy/Server as the source, and an OMNI option with S/T-omIndex set to the value included in the RS. The OMNI option also includes a Preflen confirmation, Multilink Forwarding Parameters and any other necessary OMNI sub-options. The RA also includes any information for the link, including RA Cur Hop Limit, M and O flags, Router Lifetime, Reachable Time and Retrans Timer values, and includes any necessary options such as PIOs with (A; L=0) that include MSPs for the link [RFC8028] or RIOs [RFC4191] with more-specific routes.

The FHS Proxy/Server proxies the RA using OAL encapsulation with an Identification value selected per Section 6.6, with source set to its own ADM-ULA and destination set to the MNP-ULA or temporary ULA of the Client. The FHS Proxy/Server then sends the solicited RA message to the Client and MAY later send periodic and/or event-driven unsolicited RA messages per [RFC4861]. In that case, the S/T-omIndex field in the OMNI option of each unsolicited RA message identifies the target underlying interface of the destination Client.

When the Client receives the RA message, it updates the OMNI interface NCE for the Hub Proxy/Server's ADM-LLA via the L2 address and ADM-ULA of the FHS Proxy/Server. The Client then caches the RA MFV information as the values to include in other IPv6 ND messages it sends over this underlying interface. If the Client connects to multiple *NETs, it records the additional FHS Proxy/Server L2/ADM-ULA addresses and MFV information in the Hub Proxy/Server NCE. The Client then configures default routes and assigns the Subnet Router Anycast address corresponding to the MNP (e.g., 2001:db8:1:2::) to the OMNI interface. The Client then manages its underlying interfaces according to their states as follows:

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

- When an underlying interface transitions to DOWN, the Client sends an unsolicited NA message over any UP underlying interface with an OMNI option containing Interface Attributes sub-options for the DOWN underlying 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 underlying interfaces), or may instead set the PNG flag in the OMNI header to trigger a reliable solicited NA reply.

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

- When all of a Client's underlying interfaces have transitioned to DOWN (or if the prefix registration lifetime expires), all associated Proxy/Servers withdraw the MNP the same as if they 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 underlying interface (i.e., even after attempting to contact alternate Proxy/Servers), the Client declares this underlying interface as DOWN.

The IPv6 layer sees the OMNI interface as an ordinary IPv6 interface. Therefore, when the IPv6 layer sends an RS message the OMNI interface returns an internally-generated RA message as though the message originated from an IPv6 router. The internally-generated RA message contains configuration information that is consistent with the information received from the RAs generated by the MS. Whether the OMNI interface IPv6 ND messaging process is initiated from the receipt of an RS message from the IPv6 layer or independently of the IPv6 layer is an implementation matter. Some implementations may elect to defer the IPv6 ND messaging process until an RS is received from the IPv6 layer, while others may elect to initiate the process proactively. Still other deployments may elect to administratively

disable the ordinary RS/RA messaging used by the IPv6 layer over the OMNI interface, since they are not required to drive the internal RS/RA processing. (Note that this same logic applies to IPv4 implementations that employ ICMP-based Router Discovery per [RFC1256].)

Note: Client RS messages include a Multilink Forwarding Parameters MFVI that corresponds to MFIB state that it holds for each FHS Proxy/Server used to reach the Hub, and the Hub Proxy/Server RA messages include Multilink Forwarding Parameter MFVIs that correspond to MFIB state for the Client. Each MFIB MFV entry includes both the MNP-ULA of the Client and the ADM-ULA of the Proxy/Server. Once MVF entries have been established, Clients and Proxy/Servers can exchange carrier packets using OAL header compression.

Note: The Router Lifetime value in RA messages indicates the time before which the Client must send another RS message over this underlying interface (e.g., 600 seconds), however that timescale may be significantly longer than the lifetime the MS has committed to retain the prefix registration (e.g., REACHABLETIME seconds). Proxy/Servers are therefore responsible for keeping MS state alive on a shorter timescale than the Client is required to do on its own behalf.

Note: On multicast-capable underlying 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 a unicast exchange to test reachability.

15.1. Window Synchronization

In environments where Identification window synchronization is necessary, the RS/RA exchanges discussed above observe the procedures specified in Section 6.6. The initial RS/RA exchange between a Client and Hub Proxy/Server over a first underlying interface must invoke end-to-end window synchronization when necessary, while subsequent RS/RA exchanges with the same Hub Proxy/Server performed over additional underlying interfaces within ReachableTime and with in-window Identification values need not also invoke end-to-end window synchronization. Following the initial exchange, future window (re)synchronizations can occur over any underlying interface, i.e., and not necessarily only over the one used for the initial exchange.

When a Client needs to perform window synchronization via a new FHS Proxy/Server, it sets the RS SYN source address to its own MNP-LLA

and destination address to the ADM-LLA of the Hub Proxy/Server. The Client then performs OAL encapsulation using its own MNP-ULA as the source and the ADM-ULA of the FHS Proxy/Server as the destination and includes a Multilink Forwarding Parameters sub-option with Tunnel Window Synchronization parameters then forwards the resulting carrier packets to the FHS Proxy/Server. The FHS Proxy/Server authenticates the message, caches the Tunnel Window Synchronization parameters then re-encapsulates it with its own ADM-ULA as the source and the ADM-ULA of the Hub Proxy/Server as the target.

The FHS Proxy/Server then forwards the carrier packets via the secured spanning tree to the Hub Proxy/Server, which updates its Tunnel Window Synchronization information for the FHS Proxy/Server and returns a unicast RA message with source set to its own ADM-LLA and destination set to the Client's MNP-LLA. The Hub Proxy/Server then performs OAL encapsulation using its own ADM-ULA as the source and the ADM-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 caches the Window Synchronization information, re-encapsulates the message using its own ADM-ULA as the source, the MNP-ULA of the Client as the destination, and includes an authentication signature if necessary. The FHS Proxy/Server then forwards the message to the Client which updates its window synchronization information for both the Hub and FHS Proxy/Servers as necessary.

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.).

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

For IPv6-enabled *NETs, if the underlying interface does not configure an IPv6 GUA the Client forwards the message without further

encapsulation. Otherwise, the Client encapsulates the message in UDP/IPv6 headers, sets the source to the underlying interface GUA 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.

For IPv4-only *NETs, the Client encapsulates the RS message in UDP/IPv4 headers, sets the source to the underlying interface IPv4 address and sets the destination to 192.88.99.1 (see: IANA Considerations). 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 does not configure the specified OMNI IPv6 anycast address, it should forward the OAL-encapsulated RS to another nearby Proxy/Server connected to the same IPv4 (multihop) network that does configure the 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 or another Client). 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 the Proxy/Server that configures the OMNI IPv6 anycast OAL destination receives the message, it decapsulates the RS and assumes either the FHS or Hub role, since the network layer Hop Limit is not decremented by the multihop forwarding process. The Hub Proxy/Server then prepares an RA message with source address set to its own ADM-LLA and destination address set to the Client MNP-LLA. The Hub Proxy/Server then performs OAL encapsulation and fragmentation, with OAL source set to its own ADM-ULA and destination set to the ULA corresponding to the RS source (which may be either an FHS Proxy/Server or the Client itself) and forwards to the FHS Proxy/Server if necessary. When the Hub or FHS Proxy/Server next forwards the RA to the Client, it encapsulates the message in UDP/IP headers (if necessary) with source address set to its own address and with destination set to the encapsulation source of the RS.

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.

Note: As an alternate approach to multihop forwarding via IPv6 encapsulation, the Client and Proxy/Server could statelessly translate the IPv6 LLAs into ULAs and forward the RS/RA messages without encapsulation. This would violate the [RFC4861] requirement that certain IPv6 ND messages must use link-local addresses and must not be accepted if received with Hop Limit less than 255. This document therefore mandates encapsulation since the overhead is nominal considering the infrequent nature and small size of IPv6 ND messages. Future documents may consider encapsulation avoidance through translation while updating [RFC4861].

Note: An alternate approach to multihop forwarding via IPv4 encapsulation would be to employ IPv6/IPv4 protocol translation. However, for IPv6 ND messages the LLAs would be truncated due to translation and the OMNI Router and Prefix Discovery services would not be able to function. The use of IPv4 encapsulation is therefore indicated.

15.3. DHCPv6-based Prefix Registration

When a Client is not pre-provisioned with an MNP-LLA (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 the unspecified address (::) if it has no MNP_LLAs. If the Client requires only a single MNP delegation, it can then include a Node Identification sub-option in the OMNI option and set Preflen to the length of the desired MNP. If the 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 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 the unspecified address (::) 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 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 adds routes to the routing system and creates MNP-LLAs based on the delegated MNPs. The Hub Proxy/Server then sends an RA back to the Client with the DHCPv6 Reply message included in an OMNI DHCPv6 message sub-option if and only if the RS message had included an explicit DHCPv6 Solicit. If the RS message source was the unspecified address (::), the Hub Proxy/Server includes one of the (newly-created) MNP-LLAs as the RA destination address and sets the OMNI option Preflen accordingly; otherwise, the Hub Proxy/Server includes the RS source address as the RA destination address. The Hub Proxy/Server then sets the RA source address to its own ADM-LLA then performs OAL encapsulation and fragmentation and sends the RA to the Client (i.e., either directly or via an FHS Proxy/Server). 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 as its primary MNP-LLA. The Client will then use this primary MNP-LLA as the source address of any IPv6 ND messages it sends as long as it retains ownership of the MNP.

16. Secure Redirection

If the *NET 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 *NET link, 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 *NET link 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 *NET links, 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 *NET 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 on the *NET. If a Hub Proxy/Server fails, the FHS Proxy/Server can quickly inform Clients of the outage by sending multicast RA messages on the *NET interface. The FHS Proxy/Server sends RA messages to Clients via the *NET interface with an OMNI option with a Release ID for the failed LHS Proxy/Server, and with destination address set to All-Nodes multicast (ff02::1) [RFC4291].

The FHS Proxy/Server SHOULD send MAX_FINAL_RTR_ADVERTISEMENTS RA messages separated by small delays [RFC4861]. Any Clients on the *NET interface 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 LLA source address and an OMNI option, first hop routers that recognize the OMNI 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 L2 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 underlying 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 underlying 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 underlying interface that received the RA according to standard IPv6 ND and address resolution conventions [RFC4861] [RFC4862]. Client OMNI

interfaces configured over IPv4-only underlying interfaces configure IPv4 address information on the underlying interfaces using mechanisms such as DHCPv4 [RFC2131].

Client OMNI interfaces configured over underlying 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, Client OMNI interfaces can instead send IPv6 ND messages with authentication signatures using UDP/IP encapsulation.

OMNI interfaces use UDP service port number 8060 (see: Section 25.12 and Section 3.6 of [I-D.templin-6man-aero]), and use simple UDP/IP encapsulation for both IPv4 and IPv6 underlying interfaces. The OMNI interface submits original IP packets for OAL encapsulation, then encapsulates the resulting OAL fragments immediately following a UDP header. (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.

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] 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 the IPv6 address with the neighbor (e.g., via a node identifier, MAC address, etc. embedded in the address).

Before calculating the authentication signature, the source 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 all IPv6 ND message options including all OMNI options. The source next writes the authentication signature into the sub-option signature field and forwards the message with the Checksum field still set to 0.

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 instead coordinate with other INET nodes that can provide forwarding services instead of burdening the Proxy/Server (or preferably coordinate directly with peer Clients directly). 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 underlying 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]. Proxy/Servers include Origin Indications in RA messages over INET underlying interfaces to allow Clients to detect the presence of NATs.

Note: Following the initial IPv6 ND message exchange, OMNI interfaces configured over INET underlying 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.

Note: OMNI interfaces configured over INET underlying interfaces should employ the Identification window synchronization mechanisms specified in Section 6.6 in order to reject 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.

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 source set to the unspecified 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 ULAs

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 Temporary ULA. In particular, when a Client creates an RS message it can set the source to the unspecified address (::) 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 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 a Proxy/Server receives the RS message, it notes that the source was the unspecified address (::), then examines the encapsulation source address to determine that the source is a (H)HIT and not a Temporary ULA. The Proxy/Server next invokes the DHCPv6 protocol to request an MNP prefix delegation while using the HIT (in the form of a DUID) as the Client Identifier, then prepares an RA message with

source address set to its own ADM-LLA and destination set to the MNP-LLA corresponding to the delegated MNP. The 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 IPv6 header with source address set to its own ADM-ULA and destination set to the (H)HIT from the RS encapsulation source address, then returns the encapsulated RA to the Client.

Clients can also use (H)HITs and/or Temporary ULAs 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 Temporary ULAs as original IPv6 packet source and destination addresses to support direct communications. Clients can also inject their (H)HITs and/or Temporary ULAs into a MANET/VANET routing protocol to enable multihop communications. Clients can further exchange IPv6 ND messages (such as NS/NA) using their (H)HITs and/or Temporary ULAs 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 Temporary ULAs 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 / Temporary ULA into the routing system instead of requesting an MNP prefix delegation. This would potentially enable OMNI link-wide communications using only (H)HITs or Temporary ULAs, and not MNPs. This document notes the opportunity, but makes no recommendation.

23. Address Selection

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

Clients use MNP-ULAs as source/destination IPv6 addresses in the encapsulation headers of OAL packets. Clients use Temporary ULAs for OAL addressing when an MNP-ULA is not available, or as source/destination IPv6 addresses for communications within a MANET/VANET local area. Clients can also use (H)HITs instead of Temporary ULAs 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.

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 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

The IANA is instructed to allocate an official Ethertype number TBD2 from 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. "IPv6 Neighbor Discovery Option Formats" Registry

The IANA is instructed to allocate an official Type number TBD3 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].

25.4. "Ethernet Numbers" Registry

The IANA is instructed to allocate one Ethernet unicast address TBD4 (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	Reference
00-52-14	Overlay Multilink Network (OMNI) Interface	[RFCXXXX]

Figure 32: IANA Unicast 48-bit MAC Addresses

25.5. "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 33: ICMPv6 Code Fields: Type 2 - Packet Too Big Values

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

25.6. "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	Multilink Fwding Parameters	[RFCXXXX]
3	Interface Attributes	[RFCXXXX]
4	Traffic Selector	[RFCXXXX]
5	Geo Coordinates	[RFCXXXX]
6	DHCPv6 Message	[RFCXXXX]
7	HIP Message	[RFCXXXX]
8	PIM-SM Message	[RFCXXXX]
9	Reassembly Limit	[RFCXXXX]
10	Fragmentation Report	[RFCXXXX]
11	Node Identification	[RFCXXXX]
12	ICMPv6 Error	[RFCXXXX]
13-29	Unassigned	
30	Sub-Type Extension	[RFCXXXX]
31	Reserved by IANA	[RFCXXXX]

Figure 34: OMNI Option Sub-Type Values

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

The OMNI Geo Coordinates sub-option (see: Section 12.2.6) 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]
255	Reserved by IANA	[RFCXXXX]

Figure 35: OMNI Geo Coordinates Type

25.8. "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 36: OMNI Node Identification ID-Type Values

25.9. "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 37: OMNI Option Sub-Type Extension Values

25.10. "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 38: OMNI RFC4380 UDP/IP Header Option

25.11. "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 39: OMNI RFC6081 Trailer Option

25.12. Additional Considerations

The IANA has assigned the UDP port number "8060" for an earlier experimental version of AERO [RFC6706]. This document together with [I-D.templin-6man-aero] reclaims the UDP port number "8060" for 'aero' as the service port for UDP/IP encapsulation. (Note that, although [RFC6706] was not widely implemented or deployed, any messages coded to that specification can be easily distinguished and ignored since they use an invalid ICMPv6 message type number '0'.) The IANA is therefore instructed to update the reference for UDP port number "8060" from "RFC6706" to "RFCXXXX" (i.e., this document).

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

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

The IANA is instructed to re-assign the IPv4 prefix 192.88.99.0/24 as the "OMNI IPv4 anycast" prefix. The prefix has been set aside from its former use by [RFC7526].

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.)

Client OMNI interfaces configured over secured ANET interfaces inherit the physical and/or link-layer security properties (i.e., "protected spectrum") of the connected ANETs. Client OMNI interfaces configured over open INET interfaces can use symmetric securing services such as VPNs or can by some other means establish a direct link. When a VPN or direct link may be impractical, however, the security services specified in [RFC7401] and/or [RFC4380] can be employed. While the OMNI link protects control plane messaging, applications must still employ end-to-end transport- or higher-layer security services to protect the data plane.

Strong network layer security for control plane messages and forwarding path integrity for data plane messages between 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. 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 6.10. 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 and will be presented at the June 2021 ICAO mobility subgroup meeting.

28. Document Updates

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

- [RFC1191]
- [RFC4443]
- [RFC8201]
- [RFC7526]

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: Stuart Card, Michael Matyas, Robert Moskowitz, Madhu Niraula, Greg Saccone, Stephane Tamalet, Eduard Vasilenko, Eric Vyncke. Pavel

Drasil, Zdenek Jaron and Michal Skorepa are especially recognized for their many helpful ideas and suggestions. Madhuri Madhava Badgandi, Sean Dickson, Don Dillenburg, Joe Dudkowski, Vijayasarathy Rajagopalan, Ron Sackman and Katherine Tran are acknowledged for their hard work on the implementation and technical insights that led to improvements for the spec.

Discussions on the IETF 6man and atn mailing lists during the fall of 2020 suggested additional points to consider. The authors gratefully acknowledge the list members who contributed valuable insights through those discussions. Eric Vyncke and Erik Kline were the intarea ADs, while Bob Hinden and Ole Troan were the 6man WG chairs at the time the document was developed; they are all gratefully acknowledged for their many helpful insights. Many of the ideas in this document have further built on IETF experiences beginning in the 1990s, with insights from colleagues including Ron Bonica, Brian Carpenter, Ralph Droms, Christian Huitema, Thomas Narten, Dave Thaler, Joe Touch, and many others who deserve recognition.

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

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

[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>.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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/", March 2020.

[ATN-IPS] WG-I, ICAO., "ICAO Document 9896 (Manual on the Aeronautical Telecommunication Network (ATN) using Internet Protocol Suite (IPS) Standards and Protocol), Draft Edition 3 (work-in-progress)", December 2020.

[CKSUM] Stone, J., Greenwald, M., Partridge, C., and J. Hughes, "Performance of Checksums and CRC's Over Real Data, IEEE/ ACM Transactions on Networking, Vol. 6, No. 5", October 1998.

[CRC] Jain, R., "Error Characteristics of Fiber Distributed Data Interface (FDDI), IEEE Transactions on Communications", August 1990.

[I-D.ietf-drip-rid] Moskowitz, R., Card, S. W., Wiethuechter, A., and A. Gurtov, "UAS Remote ID", draft-ietf-drip-rid-07 (work in progress), January 2021.

[I-D.ietf-intarea-tunnels] Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-10 (work in progress), September 2019.

[I-D.ietf-ipwave-vehicular-networking] (editor), J. (. J., "IPv6 Wireless Access in Vehicular Environments (IPWAVE): Problem Statement and Use Cases", draft-ietf-ipwave-vehicular-networking-20 (work in progress), March 2021.

[I-D.ietf-tsvwg-udp-options] Touch, J., "Transport Options for UDP", draft-ietf-tsvwg-udp-options-12 (work in progress), May 2021.

[I-D.templin-6man-aero] Templin, F. L., "Automatic Extended Route Optimization (AERO)", draft-templin-6man-aero-01 (work in progress), April 2021.

[I-D.templin-6man-dhcpv6-ndopt]
Templin, F. L., "A Unified Stateful/Stateless Configuration Service for IPv6", draft-templin-6man-dhcpv6-ndopt-11 (work in progress), January 2021.

[I-D.templin-6man-lla-type]
Templin, F. L., "The IPv6 Link-Local Address Type Field", draft-templin-6man-lla-type-02 (work in progress), November 2020.

[I-D.templin-6man-omni-interface]
Templin, F. L. and T. Whyman, "Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces", draft-templin-6man-omni-interface-99 (work in progress), March 2021.

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

[IPV6-GUA]
Postel, J., "IPv6 Global Unicast Address Assignments, https://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml", December 2020.

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

[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>.

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

[RFC1256] Deering, S., Ed., "ICMP Router Discovery Messages", RFC 1256, DOI 10.17487/RFC1256, September 1991, <https://www.rfc-editor.org/info/rfc1256>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <https://www.rfc-editor.org/info/rfc2131>.

[RFC2225] Laubach, M. and J. Halpern, "Classical IP and ARP over ATM", RFC 2225, DOI 10.17487/RFC2225, April 1998, <https://www.rfc-editor.org/info/rfc2225>.

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

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

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

[RFC2526] Johnson, D. and S. Deering, "Reserved IPv6 Subnet Anycast Addresses", RFC 2526, DOI 10.17487/RFC2526, March 1999, <https://www.rfc-editor.org/info/rfc2526>.

[RFC2529] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529, DOI 10.17487/RFC2529, March 1999, <https://www.rfc-editor.org/info/rfc2529>.

[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>.

[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>.

[RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, DOI 10.17487/RFC3692, January 2004, <https://www.rfc-editor.org/info/rfc3692>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <https://www.rfc-editor.org/info/rfc3810>.

[RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <https://www.rfc-editor.org/info/rfc3819>.

[RFC3879] Huitema, C. and B. Carpenter, "Deprecating Site Local Addresses", RFC 3879, DOI 10.17487/RFC3879, September 2004, <https://www.rfc-editor.org/info/rfc3879>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

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

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

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <https://www.rfc-editor.org/info/rfc4389>.

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

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

[RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, DOI 10.17487/RFC4605, August 2006, <https://www.rfc-editor.org/info/rfc4605>.

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

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <https://www.rfc-editor.org/info/rfc4963>.

[RFC5175] Haberman, B., Ed. and R. Hinden, "IPv6 Router Advertisement Flags Option", RFC 5175, DOI 10.17487/RFC5175, March 2008, <https://www.rfc-editor.org/info/rfc5175>.

[RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, DOI 10.17487/RFC5213, August 2008, <https://www.rfc-editor.org/info/rfc5213>.

[RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, DOI 10.17487/RFC5214, March 2008, <https://www.rfc-editor.org/info/rfc5214>.

[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>.

[RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <https://www.rfc-editor.org/info/rfc5798>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

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

[RFC6221] Miles, D., Ed., Ooghe, S., Dec, W., Krishnan, S., and A. Kavanagh, "Lightweight DHCPv6 Relay Agent", RFC 6221, DOI 10.17487/RFC6221, May 2011, <https://www.rfc-editor.org/info/rfc6221>.

[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>.

[RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, DOI 10.17487/RFC6355, August 2011, <https://www.rfc-editor.org/info/rfc6355>.

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

[RFC6543] Gundavelli, S., "Reserved IPv6 Interface Identifier for Proxy Mobile IPv6", RFC 6543, DOI 10.17487/RFC6543, May 2012, <https://www.rfc-editor.org/info/rfc6543>.

[RFC6706] Templin, F., Ed., "Asymmetric Extended Route Optimization (AERO)", RFC 6706, DOI 10.17487/RFC6706, August 2012, <https://www.rfc-editor.org/info/rfc6706>.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <https://www.rfc-editor.org/info/rfc6935>.

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

[RFC6980] Gont, F., "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery", RFC 6980, DOI 10.17487/RFC6980, August 2013, <https://www.rfc-editor.org/info/rfc6980>.

[RFC7042] Eastlake 3rd, D. and J. Abley, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", BCP 141, RFC 7042, DOI 10.17487/RFC7042, October 2013, <https://www.rfc-editor.org/info/rfc7042>.

[RFC7084] Singh, H., Beebee, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, DOI 10.17487/RFC7084, November 2013, <https://www.rfc-editor.org/info/rfc7084>.

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <https://www.rfc-editor.org/info/rfc7323>.

[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>.

[RFC7526] Troan, O. and B. Carpenter, Ed., "Deprecating the Anycast Prefix for 6to4 Relay Routers", BCP 196, RFC 7526, DOI 10.17487/RFC7526, May 2015, <https://www.rfc-editor.org/info/rfc7526>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <https://www.rfc-editor.org/info/rfc7542>.

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

[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>.

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

[RFC8892] Thaler, D. and D. Romascanu, "Guidelines and Registration Procedures for Interface Types and Tunnel Types", RFC 8892, DOI 10.17487/RFC8892, August 2020, <https://www.rfc-editor.org/info/rfc8892>.

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

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

[RFC8981] Gont, F., Krishnan, S., Narten, T., and R. Draves, "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6", RFC 8981, DOI 10.17487/RFC8981, February 2021, <https://www.rfc-editor.org/info/rfc8981>.

Appendix A. 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, the encapsulated IP packet and the two-octet trailing checksum field initialized to 0. 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 as data octets D[41] through D[N-2] and finally concludes with the two trailing 0 octets as data octets D[N-1] and 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. However, OMNI interfaces omit unnecessarily redundant checks to improve performance and minimize complexity.

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 ANET interface, it need not include an authentication sub-option but instead calculates/includes an IPv6 ND message checksum. The OMNI interface that receives the message applies any lower-layer ANET authentication and integrity checks, then verifies the OAL checksum (if present) followed by the IPv6 ND message checksum.

When an OMNI interface sends NS/NA(NUD) messages that do not traverse the secured spanning tree, it includes an authentication option only if authentication is necessary; otherwise, it calculates/includes the IPv6 ND message checksum.

When a FHS Proxy/Server forwards a proxyed IPv6 ND message into the secured spanning tree, it omits both the authentication sub-option and IPv6 ND message checksum (i.e., even if it alters the IPv6 ND message contents before forwarding) since the secured spanning tree assures authentication and integrity through lower-layer security services. The OMNI interface that receives the message has assurance that authentication and integrity are protected by lower layers.

OAL destinations discard carrier packets with unacceptable Identifications and submit the encapsulated fragments in 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 L2 Address Mapping

Per [RFC4861], IPv6 ND messages may be sent to either a multicast or unicast link-scoped IPv6 destination address. However, IPv6 ND messaging should be coordinated between the Client and Proxy/Server only without invoking other nodes on the *NET. 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 *NET links, Proxy/Servers can maintain an OMNI-specific unicast L2 address ("MSADDR"). For Ethernet-compatible *NETs, this specification reserves one Ethernet unicast address TBD4 (see: Section 25). For non-Ethernet statically-addressed *NETs, MSADDR is reserved per the assigned numbers authority for the *NET addressing space. For still other

*NETs, MSADDR may be dynamically discovered through other means, e.g., L2 beacons.

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 L2 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 draft-templin-6man-omni-32 to draft-templin-6man-omni-33:

o That's it, folks.

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

o Only one FHS Proxy/Server is elected as the Hub, and only the Hub provides designated router and mobility anchor point services.

o Re-adjusted OMNI sub-options to separate Interface Attributes from Traffic Selectors.

o Removed MS-Register/Release.

o Anycast.

Differences from draft-templin-6man-omni-30 to draft-templin-6man-omni-31:

o Major changes, especially in Sections 6.2, 6.4, 6.5, 12.2.15 and others.

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

o Major revision update for review.

Differences from draft-templin-6man-omni-28 to draft-templin-6man-omni-29:

o Interim version with extensive new text - cleanup planned for next release.

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

o Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

Differences from draft-templin-6man-omni-26 to draft-templin-6man-omni-27:

o Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

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

o Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

Differences from draft-templin-6man-omni-24 to draft-templin-6man-omni-25:

o Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

Differences from draft-templin-6man-omni-23 to draft-templin-6man-omni-24:

o Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

Differences from draft-templin-6man-omni-22 to draft-templin-6man-omni-23:

o Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

Differences from draft-templin-6man-omni-21 to draft-templin-6man-omni-22:

- Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

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

- Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

Differences from draft-templin-6man-omni-19 to draft-templin-6man-omni-20:

- Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

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

- Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

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

- Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

Differences from draft-templin-6man-omni-16 to draft-templin-6man-omni-17:

- Final editorial review pass resulting in multiple changes. Document now submit for final approval (with reference to rfcdiff from previous version).

Differences from draft-templin-6man-omni-15 to draft-templin-6man-omni-16:

- Final editorial review pass resulting in multiple changes. Document now submit for final approval.

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

- o Text restructuring to remove ambiguities, eliminate extraneous text and improve readability.

- o Clarified that the OMNI link model is NBMA and that link-scoped multicast is through iterative unicast.

Differences from draft-templin-6man-omni-13 to draft-templin-6man-omni-14:

- o Brought back the optional two-message exchange feature.

- o Added TCP RST flag and new (OPT, PNG) flags to the OMNI option header.

- o Require the OAL node that initiates the symmetric connection to include its (future) receive window size in the initial SYN.

- o Require OAL nodes to select new ISS values that are outside of the current SND.WND.

- o Text clarifications for improved readability.

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

- o Complete revision of OAL Identification Window Maintenance section to incorporate well-known protocol conventions and terminology.

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

- o Expanded on details of symmetric window synchronization.

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

- o Included an Ordinal Number field in the Compressed Header format for non-final fragments

- o Clarified that the window coordination protocol is based on the IPv6 ND connectionless protocol using TCP constructs, and not based on the TCP connection-oriented protocol.

- o Removed unneeded fields from the OMNI option header.

Differences from draft-templin-6man-omni-09 to draft-templin-6man-omni-10:

- o Fixed sizing considerations for OMNI option fields.

- o Updated handling of multiple OMNI options in the same IPv6 ND message. Only the first option includes the header, while all other options include only sub-options.

Differences from draft-templin-6man-omni-08 to draft-templin-6man-omni-09:

- o Included reference to RFC3366 and updated section on Fragment Retransmission.

- o Added "ordinal number" marking in Fragment Header reserved field.

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

- o Included TCP state variables; window scale

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

- o Moved Interface Attributes, Type 1 and Type 2 to historic status.

- o Incorporated Traffic Selector into Interface Attributes, Type 4.

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

- o Adopted TCP as an OAL packet-based connection-oriented protocol.

- o Three-Way handshake for establishing symmetric send/receive windows

- o Window length specified, plus "current" and "previous" windows

- o New appendix on checksum algorithm, with citations changed

- o Security architecture considerations.

- o More details on HIP message signatures.

- o Require firewalls at OAL destinations.

- o Removed "equal-length" requirement for OAL non-final fragments.

Differences from draft-templin-6man-omni-04 to draft-templin-6man-omni-05:

- Change to S/T-omIndex definition.

Differences from draft-templin-6man-omni-03 to draft-templin-6man-omni-04:

- Changed reference citations to "draft-templin-6man-aero".
- Included introductory description of the "6M's".
- Included new OMNI sub-option for PIM-SM.

Differences from draft-templin-6man-omni-02 to draft-templin-6man-omni-03:

- Added citation of RFC8726.

Differences from draft-templin-6man-omni-01 to draft-templin-6man-omni-02:

- Updated IANA registration policies for OMNI registries.

Differences from draft-templin-6man-omni-00 to draft-templin-6man-omni-01:

- Changed intended document status to Informational, and removed documents from "updates" category.
- Updated implementation status.
- Minor edits to HIP message specifications.
- Clarified OAL and *NET IP header field settings during encapsulation and re-encapsulation.

Differences from earlier versions to draft-templin-6man-omni-00:

- Established working baseline reference.

Authors' Addresses

Fred L. Templin (editor)
The Boeing Company
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

Tony Whyman
MWA Ltd c/o Inmarsat Global Ltd
99 City Road
London EC1Y 1AX
England

Email: tony.whyman@mccallumwhyman.com