Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces
draft-templin-6man-omni-24

Abstract

   Mobile nodes (e.g., aircraft of various configurations, terrestrial
   vehicles, seagoing vessels, enterprise wireless devices, etc.)
   communicate with networked correspondents over multiple access
   network data links and configure mobile routers to connect end user
   networks. A multilink interface specification is presented that
   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.

Status of This Memo

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

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

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

   This Internet-Draft will expire on December 9, 2021.

Copyright Notice

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 to the network-based mobility
   service to coordinate its data links in a discipline known as
   "multilink", in which a single virtual interface is configured over
   the Client's underlying interface data link connections.

   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.

   Each Client receives a Mobile Network Prefix (MNP) through mobility
   service control message exchanges with Proxy/Servers which also
   configure OMNI interfaces. 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 router discovery process
   over each of the OMNI interface's underlying interfaces in order to
   register per-link parameters (see Section 15).

   The OMNI interface provides a multilink nexus for exchanging inbound
   and outbound traffic via the correct underlying interface(s). The IP
   layer sees the OMNI interface as a point of connection to the OMNI
   link. Each OMNI link has one or more associated Mobility Service
   Prefixes (MSPs), which are typically IP Global Unicast Address (GUA)
   prefixes from which MNPs are derived. If there are multiple OMNI
   links, the IPv6 layer will see multiple OMNI interfaces.

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, and the OMNI interface then applies Performance-Based
Multilink (PBM) internally to select appropriate underlying
interfaces. Applications can apply Segment Routing [RFC8402] to
select independent SBM topologies for fault tolerance, while the OMNI
interface orchestrates PBM.

The OMNI interface interacts with a network-based Mobility Service
(MS) and/or other Clients through IPv6 Neighbor Discovery (ND)
control message exchanges [RFC4861]. The MS provides includes Proxy/
Servers (and other infrastructure elements) that track Client
movements and represent their MNPs in a global routing or mapping
system. An example MS appears (along with definitions of its other
required mobility service elements) appears in
[I-D.templin-6man-aero].

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

In addition to many other aspects, 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 in the data plane, while using IPv6 ND messaging as the
control plane independently of the data plane IP protocol(s). The
OAL operates as a sublayer between L3 and L2 based on IPv6
encapsulation [RFC2473] as discussed in the following sections.

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

Proxy/Server
   a router that provides an entry point into the MS and coordinates
   Client mobility events. 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 forwards
   ordinary data packets between OMNI interface Clients and networked
   correspondent nodes.

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.

   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.

   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.
      Multilinking decisions 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 underlying interface(s) for packet
    transmission and reception within a single OMNI interface.

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

3. Requirements

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

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

4. Overlay Multilink Network (OMNI) Interface Model

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

```
                        +----------------------------+
                        |     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  |
                        |      |      |        |     |
                        +------+------+        +------+
```
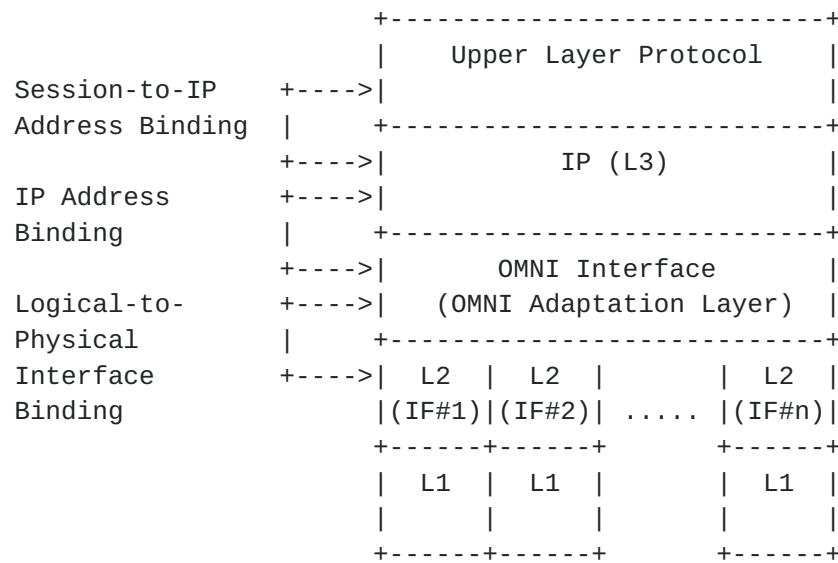
         Figure 1: OMNI Interface Architectural Layering Model

   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 are seen
as "bridges" capable of multinet concatenation.

The OMNI interface can send/receive original IP packets to/from
underlying interfaces while including/omitting various encapsulations
including OAL, UDP, IP and L2. The network layer can also access the
underlying interfaces directly while bypassing the OMNI interface
entirely when necessary. This architectural flexibility may be
beneficial for underlying interfaces (e.g., some aviation data links)
for which encapsulation overhead may be a primary consideration.
OMNI interfaces that send original IP packets directly over
underlying interfaces without invoking the OAL can only reach peers
located on the same OMNI link segment. However, when a source Client
sends carrier packets to a target Client located in a foreign network
administrative domain, it can send them to a First-Hop Segment (FHS)
Proxy/Server. The FHS Proxy/Server then forwards the carrier packets
into the Segment Routing Topology (SRT), which transport them to a
Last-Hop Segment (LHS) Proxy/Server. The LHS Proxy/Server finally
forwards the carrier packets to 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), OMNI/IP,
OMNI/UDP/IP, OMNI/UDP/IP/L2, OMNI/OAL/UDP/IP, OMNI/OAL/UDP/L2, etc.

```
+---------------------------------------------------------------+
|                      Network Layer                            |
+--+------------------------------------------------------------+
   |                     OMNI Interface                         |
   +-------------------------+-----------------------------------+
                             |      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                   |
   +---------------------------------------------------------------+
```
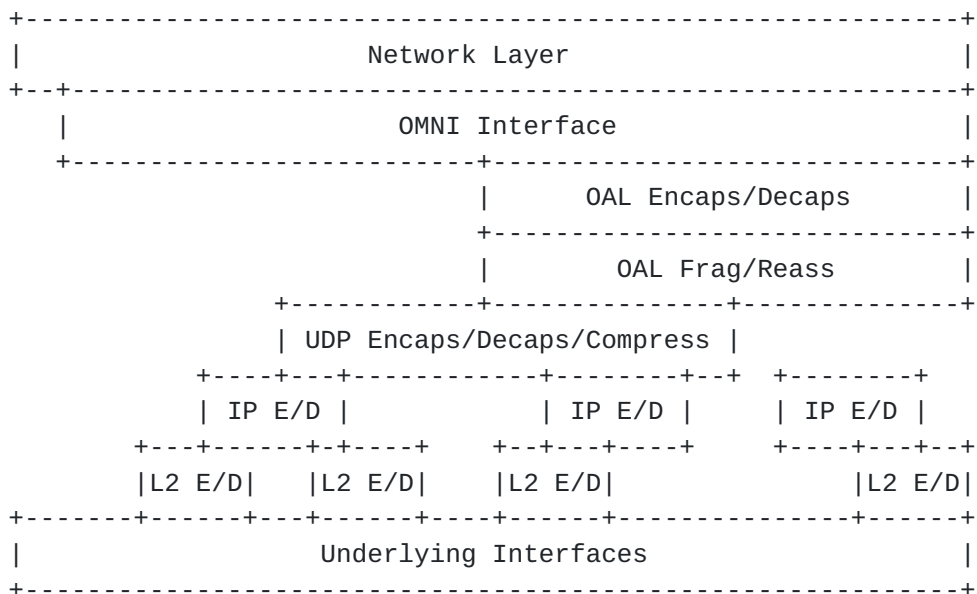
Figure 2: OMNI Interface Layering

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.

Other opportunities are discussed in [RFC7847]. Note that even when
the OMNI virtual interface is present, applications can still access
underlying interfaces either through the network protocol stack using
an Internet socket or directly using a raw socket. This allows for
intra-network (or point-to-point) communications without invoking the
OMNI interface and/or OAL. For example, when an IPv6 OMNI interface
is configured over an underlying IPv4 interface, applications can
still invoke IPv4 intra-network communications as long as the
communicating endpoints are not subject to mobility dynamics.

Figure 3 depicts the architectural model for a source Client with an
attached EUN connecting to the OMNI link via multiple independent
*NETs. When an underlying interface becomes active, the Client's
OMNI interface sends IPv6 ND messages to a FHS Proxy/Server 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
proxied 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 interface appears to be attached to an
ordinary 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 associated with the same local Proxy/

Server, the FHS and LHS Proxy/Servers are one and the same.) Clients
therefore use Proxy/Servers as gateways into the SRT to reach distant
OMNI link correspondents. When route optimization is applied as
discussed in [I-D.templin-6man-aero], Clients can instead forward
directly into the SRT (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 6.9.

The OMNI interface configures an MTU and MRU of 9180 bytes [RFC2492];
the size is therefore not a reflection of the underlying interface or
*NET path MTUs, but rather determines the largest original IP packet
the OAL (and/or underlying interface) can forward or reassemble. For
each OAL destination (i.e., for each OMNI link neighbor), the OAL
source may discover "hard" or "soft" Reassembly Limit values smaller

than the MRU based on receipt of IPv6 ND messages with OMNI
Reassembly Limit sub-options (see: Section 12.2.11). The OMNI
interface employs the OAL as an encapsulation sublayer service to
transform original IP packets into OAL packets/fragments, and the OAL
in turn uses *NET encapsulation to forward carrier packets over the
underlying interfaces (see: Section 6).

6. The OMNI Adaptation Layer (OAL)

When an OMNI interface forwards an original IP packet from the
network layer for transmission over one or more underlying
interfaces, the OMNI Adaptation Layer (OAL) acting as the OAL source
drops the packet and returns a PTB message if the packet exceeds the
MRU and/or the hard Reassembly Limit for the intended OAL
destination. Otherwise, the OAL source applies encapsulation to form
OAL packets 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 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|
+----------+-----+-----+-----+-----+-----+-----+----+
```
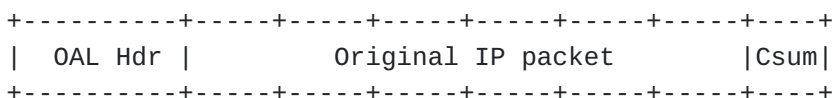
Figure 4: OAL Packet Before Fragmentation

The OAL source next selects a 32-bit Identification value for the
packet as specified in Section 6.5 then calculates the checksum per
the 8-bit Fletcher 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 writes the results
over the (0-initialized) trailing checksum octets. The OAL source
then inserts a single OMNI Routing Header (ORH) if necessary (see:
[I-D.templin-6man-aero]) while incrementing Payload Length to reflect

the addition of the ORH, where the late addition of the ORH is not
covered by the checksum. (Alternatively, the OAL source can defer
ORH insertion until after fragmentation, then manually insert an
identical copy of the ORH between the IPv6 header and Fragment Header
of each fragment while resetting the IPv6 Payload Length and Next
Header fields accordingly.)

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. In the
worst case, each successive SRT segment may re-encapsulate with
either a 20 byte IPv4 or 40 byte IPv6 header, an 8 byte UDP header
and in some cases an IP security encapsulation (40 bytes maximum
assumed). Forwarding nodes at any SRT segment may also insert a
maximum-length (40 byte) ORH as an extension to the existing 40 byte
OAL IPv6 header plus 8 byte Fragment Header if an ORH was not already
present. Assuming therefore an absolute worst case of (40 + 40 + 8)
= 88 bytes for *NET encapsulation plus (40 + 40 + 8) = 88 bytes for
OAL encapsulation leaves (576 - 88 - 88) = 400 bytes to accommodate a
portion of the original IP packet/fragment. The OAL source therefore
sets a minimum Maximum Payload Size (MPS) of 400 bytes as the basis
for the minimum-sized OAL fragment that can be assured of traversing
all 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 base path MPS on this larger size (i.e., instead of 576
bytes) as long as the *NET peer reassembles before re-encapsulating
and forwarding (while re-fragmenting if necessary). Also, if the OAL
source has a way of knowing the maximum *NET encapsulation size for
all 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 a full header could be included 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 message in response (with the
possible receipt of link-layer error message in case the probe was
lost). For this purpose, the OAL source can send an NS message with
one or more OMNI options with large PadN sub-options (see:
Section 12) in order to receive a small NA response from the OAL
destination. While observing the minimum MPS will always result in
robust and secure behavior, the OAL source should optimize path MPS
values when more efficient utilization may result in better
performance (e.g. for wireless aviation data links). (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.
In particular, 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, at most 23 fragments will be
produced for each OAL packet.

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 link persistence period in case link-layer
retransmission is requested (see: Section 6.6). The formats of OAL
fragments and carrier packets 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
```
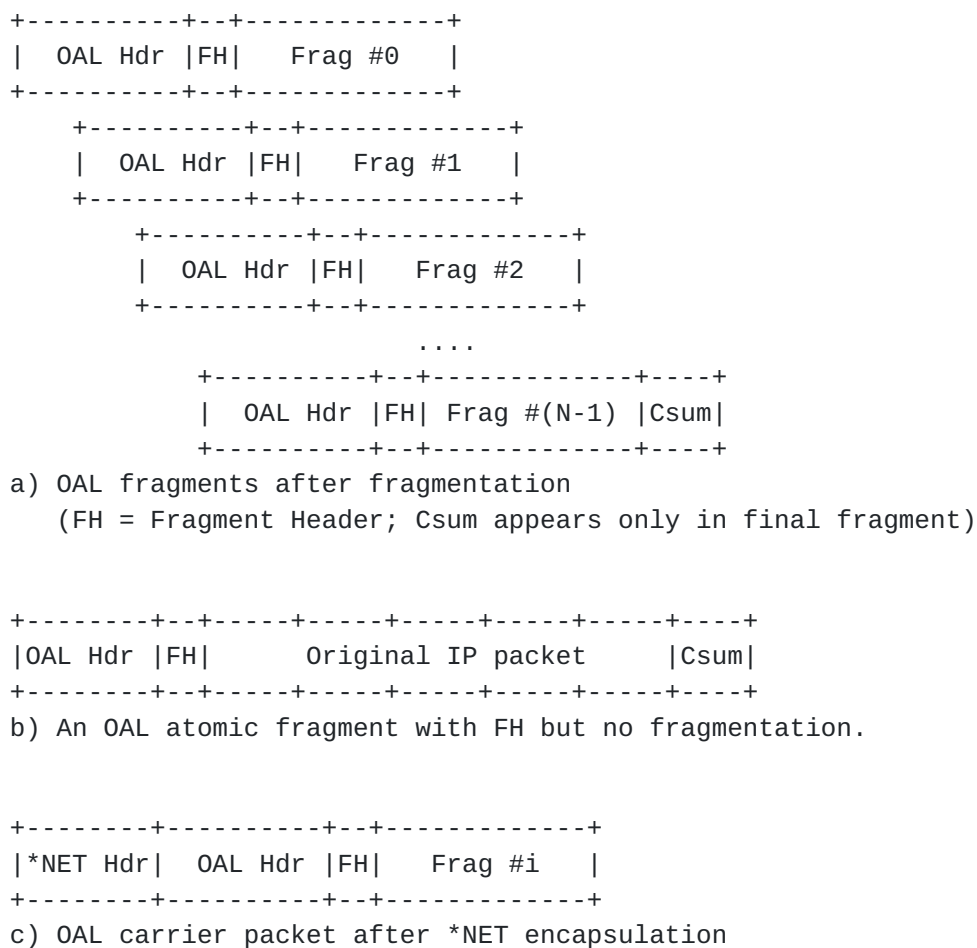
                Figure 6: OAL Fragments and Carrier Packets

6.2. OAL *NET Encapsulation and Re-Encapsulation

During *NET encapsulation, the OAL source first encapsulates each OAL
fragment in a UDP header as the first *NET encapsulation sublayer if
NAT traversal, packet filtering middlebox traversal and/or OAL header
compression are necessary. The [RFC2473] encapsulation header
appears immediately following the UDP header with the IPv6 protocol
version '6' coded in the first four bits (the first four bits may
encode different values to identify other encapsulation formats
discussed later in this document). The OAL source then appends any

additional encapsulation sublayer headers necessary and presents the
*NET packet to an underlying interface (see: Figure 2).

When a UDP header is included, the OAL source next sets the UDP
source port to a constant value that it will use in each successive
carrier packet it sends to the next OAL hop. For packets sent to a
FHS Proxy/Server, the OAL source sets the UDP destination port to
8060, i.e., the IANA-registered port number for AERO. For packets
sent to a peer Client, the source sets the UDP destination port to
the cached port value for this peer. The OAL source then sets the
UDP length to the total length of the OAL fragment in correspondence
with the OAL header Payload Length (i.e., the UDP length and IPv6
Payload Length must agree). The OAL source finally sets the UDP
checksum to 0 [RFC6935][RFC6936] since the only fields not already
covered by the OAL checksum or underlying *NET CRCs are the Fragment
Header fields, and any corruption in those fields will be garbage
collected by the reassembly algorithm (however, see Section 20 for
additional considerations). The UDP encapsulation header is often
used in association with IP encapsulation, but may also be used
between neighbors on a shared physical link with a true L2 header
format such as for transmission over IEEE 802 Ethernet links. This
document therefore requests a new Ether Type code assignment TBD1 in
the IANA 'ieee-802-numbers' registry for direct User Datagram
Protocol (UDP) encapsulation over IEEE 802 Ethernet links (see:
Section 25).

For *NET encapsulations over IP, the OAL source 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) sets the *NET IPv6 header
"Flow Label" as specified in [RFC6438]. The OAL source then sets the
*NET IP TTL/Hop Limit the same as for any *NET host, i.e., it does
not copy the Hop Limit value from the OAL header. For carrier
packets undergoing OAL intermediate node re-encapsulation, the node
decrements the OAL IPv6 header Hop Limit and discards the carrier
packet if the value reaches 0. The node then copies the "Type of
Service/Traffic Class" and "Congestion Experienced" values from the
previous hop *NET encapsulation header into the OAL IPv6 header
before setting the next hop *NET IP encapsulation header values the
same as specified for the OAL source above.

Following *NET encapsulation/re-encapsulation, the OAL source sends
the resulting carrier packets over one or more underlying interfaces.
The underlying interfaces often connect directly to physical media on
the local platform (e.g., a laptop computer with WiFi, etc.), but in
some configurations the physical media may be hosted on a separate
Local Area Network (LAN) node. In that case, the OMNI interface can
establish a Layer-2 VLAN or a point-to-point tunnel (at a layer below

the underlying interface) to the node hosting the physical media.
The OMNI interface may also apply encapsulation at the underlying
interface layer (e.g., as for a tunnel virtual interface) such that
carrier packets would appear "double-encapsulated" on the LAN; the
node hosting the physical media in turn removes the LAN encapsulation
prior to transmission or inserts it following reception. Finally,
the underlying interface must monitor the node hosting the physical
media (e.g., through periodic keepalives) so that it can convey
up/down/status information to the OMNI interface.

6.3. OAL Destination Decapsulation and Reassembly

When an OMNI interface receives a carrier packet from an underlying
interface, 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 below.
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. 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 records the ordinal number of each accepted
fragment of the same OAL packet (i.e., as Frag #0, Frag #1, Frag #2,
etc.) and admits them into the reassembly cache.

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

6.4. OAL Header Compression

When the OAL source and destination are on the same SRT segment,
carrier packet header compression is possible. When the OAL source
and destination exchange IPv6 ND messages, each caches the observed
*NET UDP source port and source IP (or L2) address associated with
the OAL IPv6 source address found in the full-length OAL IPv6 header.
After the initial IPv6 ND message exchange, the OAL source can apply
OAL Header Compression for subsequent carrier packets to
significantly reduce encapsulation overhead.

When the OAL source uses INET encapsulation to send carrier packets
directly to an OAL destination with NCE state, it can begin omitting
significant portions of the IPv6 header, Fragment Header and OMNI
Routing Header (ORH). (Conversely, the OAL source must still include
full headers for destinations that can only be reached via an OAL
intermediate node.) For OAL first-fragments (including atomic
fragments), the OAL source 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ *
|          Source port          |       Destination port        | U
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ D
|            Length             |           Checksum            | P
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ *
|Vers=0 | Traffic Class |            Flow Label                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |  Hop Limit  |M|      Identification (0 -1)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Identification (2-3)    |    omIndex    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

In this format, the UDP header appears in its entirety in the first 8
octets and is followed by a compressed IPv6 header with Version set
to 0 (to distinguish OCH-0 from both OCH-1 and a true IP protocol
version number), followed by the uncompressed (Traffic Class, Flow
Label, Next Header) fields and a 7-bit compressed Hop Limit that
encodes the minimum of the uncompressed Hop Limit and 127. (Note:
the OAL source sets Next Header to the protocol number of the header
following the final extension header, and not to the protocol number
for the extension header itself.) The compressed IPv6 header 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 followed by a

compressed ORH consisting of a 1-octet omIndex that encodes an
underlying interface index for the target Client (or 0 if the target
underlying interface is unspecified). The OCH-0 header is then
followed by the OAL fragment body, and the UDP length field is
reduced by the difference in length between the compressed headers
and full-length (IPv6, Fragment, ORH) headers. The OCH-0 format
applies only for first fragments, which are always regarded as
ordinal fragment 0 even though the OCH-0 does not include an explicit
Ordinal field.

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ *
|          Source port          |       Destination port        | U
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ D
|            Length             |            Checksum            | P
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ *
|Vers=1 | Ordinal |R|M|    Fragment Offset      |    ID (0)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Identification (1-3)              |    omIndex     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
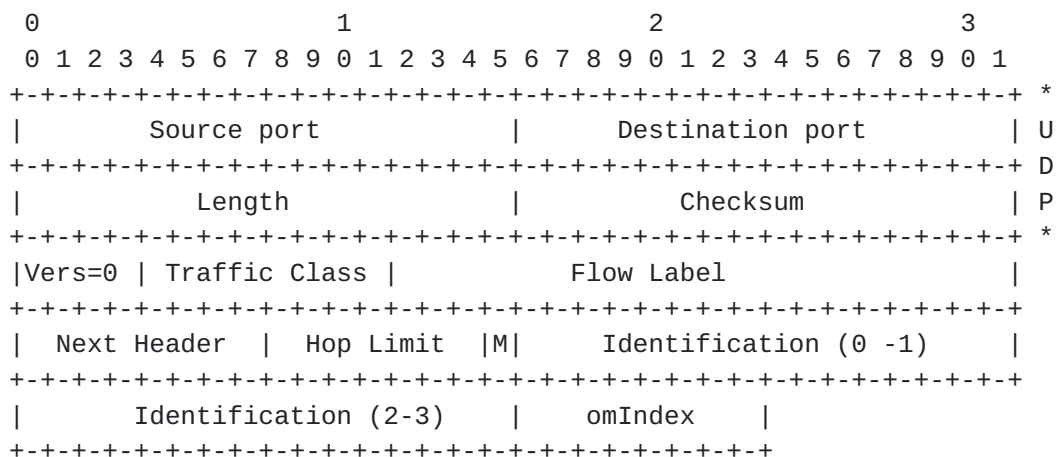
              Figure 8: OMNI Compressed Header - Type 1 (OCH-1)

In this format, the UDP header appears in its entirety in the first 8
octets, but all IPv6 header fields except for Version are omitted.
Version is set to 1 (to distinguish OCH-1 from both OCH-0 and a true
IP protocol version number) and is followed by a compressed IPv6
Fragment Header that includes a 5-bit Ordinal number for this
fragment. a (R)eserved bit set to 0, and with (M)ore Fragments/
Fragment Offset/Identification copied from the uncompressed fragment
header. The compressed ORH includes a 1-octet omIndex that encodes
an underlying interface index for the target Client (or 0 if the
target underlying interface is unspecified). The OCH-1 header is
then followed by the OAL fragment body, while the UDP length field is
reduced by the difference in length between the compressed headers
and full-length (IPv6, Fragment, ORH) headers. The OCH-1 format
applies only for non-first fragments, therefore Ordinal is set to a
value beginning with 1 for the first non-first fragment and
monotonically incremented for each successive non-first fragment up
to and including the final fragment.

When the OAL destination receives a carrier packet with an OCH, it
first determines the OAL IPv6 source and destination addresses by

examining the UDP source port and L2 source address, then determines the length by examining the UDP length. The OAL destination then examines the Version field immediately following the UDP header. If Version encodes the value 0, the OAL destination processes the remainder of the header as an OCH-0, then reconstitutes the full-sized IPv6 and Fragment Headers and adds this OAL fragment to the reassembly buffer if the fragment is acceptable. If Version encodes the value 1, the OAL destination instead processes the remainder of the header as an OCH-1, then reconstitutes the full-sized IPv6 and Fragment Headers. Note that, since OCH-1 does not include Traffic Class, Flow Label, Next Header or Hop Limit information, the OAL destination writes the value 0 into those fields when it reconstitutes the full headers. The values will be correctly populated during reassembly after an OAL first fragment with an OCH-0 or uncompressed OAL header arrives.

When the OAL destination is a Proxy/Server, it examines the destination address after re-constituting the OAL header. If the destination address is its own ADM-ULA, the Proxy/Server submits the resulting OAL fragment for local reassembly. Following reassembly, the Proxy/Server re-encapsulates the OAL packet (while re-fragmenting if necessary) and forwards the packet/fragments either into the SRT for remote targets, or to the underlying interface identified by omIndex for local targets. If the header compression state and/or destination address are not recognized, the Proxy/Server instead drops the packet.

When the OAL destination is the Client, it examines the destination address after re-constituting the OAL header. If the destination address is its own MNP-ULA, the Client submits the resulting OAL fragment for local reassembly. Otherwise, the Client drops the packet.

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. Carrier packets may further include uncompressed headers at any time even after header compression state has been established.

6.5. 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 can send OAL packets beginning with a random initial Identification value and incremented (modulo 2**32) for each successive packet. 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 maintain windows using
TCP-like synchronization [RFC0793] with Identification sequence
numbers beginning with an unpredictable initial value [RFC7739] and
incremented (modulo 2 *32) for each successive OAL packet.

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).

   o 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.

   o 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 12.2.12).

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

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

o 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.

o 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.

o 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 include authentication signatures in IPv6 ND
   messages while using unpredictable Identification values until
   window 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.

6.6. OAL Fragment Retransmission

When the OAL source sends carrier packets to an OAL destination, it
should cache recently sent packets in case 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 send a uNA message to the OAL source. The
OAL destination creates a uNA message with an OMNI option containing
an authentication sub-option to provide authentication (if the OAL
source is on an open Internetwork) and 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 signs the message if an
authentication sub-option is included, performs OAL encapsulation
(with the its own address as the OAL source and the source address of
the message that prompted the uNA as the OAL destination) and sends
the message to the OAL source.

When the OAL source receives the uNA message, it authenticates the
message using the authentication sub-option (if present) then
examines the Fragmentation Report. For each (Source, Destination,
Identification)-tuple, the OAL source determines whether it still
holds the 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 and no others. 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 repeat the request in a small number of additional uNAs within
the link persistence timeframe.

The OAL therefore provides a link-layer low persistence Automatic
Repeat Request (ARQ) service based on Selective Repeat (SR)
capability 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.

6.7. OAL MTU Feedback Messaging

When the OMNI interface forwards original IP packets from the network
layer, it invokes the OAL and returns internally-generated ICMPv4
Fragmentation Needed [RFC1191] or ICMPv6 Path MTU Discovery (PMTUD)
Packet Too Big (PTB) [RFC8201] messages as necessary. This document
refers to both of these ICMPv4/ICMPv6 message types simply as "PTBs",
and introduces a distinction between PTB "hard" and "soft" errors as
discussed below.

Ordinary PTB messages with ICMPv4 header "unused" field or ICMPv6
header Code field value 0 are hard errors that always indicate that a
packet has been dropped due to a real MTU restriction. In
particular, the OAL source drops the packet and returns a PTB hard
error if the packet exceeds the OAL destination MRU. However, the
OMNI interface can also forward large original IP packets via OAL
encapsulation and fragmentation while at the same time returning PTB
soft error messages (subject to rate limiting) if it deems the
original IP packet too large according to factors such as link
performance characteristics, reassembly congestion, etc. This
ensures that the path MTU is adaptive and reflects the current path
used for a given data flow. The OMNI interface can therefore
continuously forward packets without loss while returning PTB soft
error messages recommending a smaller size if necessary. Original
sources that receive the soft errors in turn reduce the size of the
packets they send (i.e., the same as for hard errors), but can soon
resume sending larger packets if the soft errors subside.

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

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.

6.8. OAL Requirements

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

o OAL sources MUST NOT send OAL fragments including original IP
   packets larger than the OMNI interface MTU or the OAL destination
   hard Reassembly Limit, i.e., whether or not fragmentation is
   needed.

o OAL sources MUST NOT fragment original IP packets smaller than the
   minimum MPS minus the trailer size, but must instead encapsulate
   them as atomic fragments.

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

o OAL sources MUST NOT send OAL fragments that include any extension
   headers other than a single ORH and a single Fragment Header.

o OAL intermediate nodes SHOULD and OAL destinations MUST
   unconditionally drop 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.

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

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

o OAL 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 6.7) and decrease the delay when reassembly congestion subsides.

6.9. OAL Fragmentation Security Implications

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

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

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

3. Attacks based on predictable fragment identification values - in environments where spoofing is possible, this threat is mitigated through the use of Identification windows per Section 6.5. By maintaining windows of acceptable Identifications beginning with unpredictable values, 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, with the OAL destination checksum providing a last-resort integrity verification.) 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 current 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.

6.10. OAL Super-Packets

By default, the OAL source includes a 40-byte IPv6 encapsulation header for each original IP packet during OAL encapsulation. The OAL source also calculates and appends a 2 octet trailing checksum then performs fragmentation such that a copy of the 40-byte IPv6 header plus an 8-byte IPv6 Fragment Header is included in each OAL fragment (when an ORH is added, the OAL encapsulation headers become larger still). However, these encapsulations may represent excessive overhead in some environments. OAL header compression can

   dramatically reduce the amount of encapsulation overhead, however a
   complimentary technique known as "packing" (see:
   [I-D.ietf-intarea-tunnels]) is also supported so that multiple
   original IP packets and/or control messages can be included within a
   single OAL "super-packet".

   When the OAL source has multiple original IP packets to send to the
   same OAL destination with total length no larger than the OAL
   destination MRU, it can concatenate them into a super-packet
   encapsulated in a single OAL header and trailing checksum. Within
   the OAL super-packet, the IP header of the first original IP packet
   (iHa) followed by its data (iDa) is concatenated immediately
   following the OAL header, then the IP header of the next original
   packet (iHb) followed by its data (iDb) is concatenated immediately
   following the first original packet, etc. with the trailing checksum
   included last. The OAL super-packet format is transposed from
   [I-D.ietf-intarea-tunnels] and shown in Figure 9:

```
                     <------- 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 --->
```
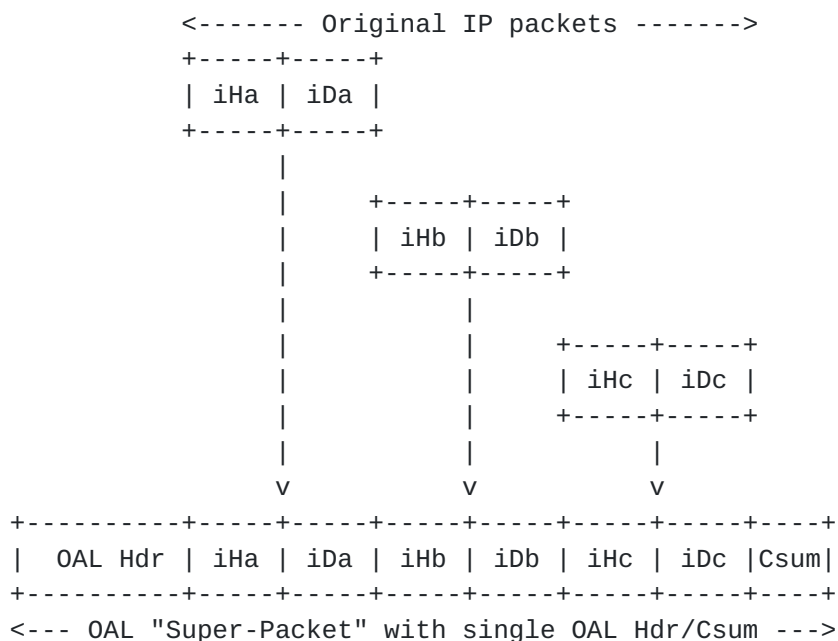
                    Figure 9: OAL Super-Packet Format

   When the OAL source prepares a super-packet, it applies OAL
   fragmentation and *NET encapsulation then sends the carrier packets
   to the OAL destination. When the OAL destination receives the super-
   packet it reassembles if necessary, verifies and removes the trailing
   checksum, then regards the remaining OAL header Payload Length as the
   sum of the lengths of all payload packets. The OAL destination then
   selectively extracts each original IP packet (e.g., by setting
   pointers into the super-packet buffer and maintaining a reference
   count, by copying each packet into a separate buffer, etc.) and

forwards each packet to the network layer. During extraction, the
OAL determines the IP protocol version of each successive original IP
packet 'j' by examining the four most-significant bits of iH(j), and
determines the length of the packet by examining the rest of iH(j)
according to the IP protocol version.

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 is fe80::2001:db8:1000:2000/120. Non-
   MNP routes are also represented the same as for MNP-LLAs, but
   include a GUA prefix that is not properly covered by the MSP.

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

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

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

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

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

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

9. Unique-Local Addresses (ULAs)

OMNI domains use IPv6 Unique-Local Addresses (ULAs) as the source and
destination addresses in OAL packet IPv6 encapsulation headers. ULAs
are only routable within the scope of a an OMNI domain, and are
derived from the IPv6 Unique Local Address prefix fc00::/7 followed
by the L bit set to 1 (i.e., as fd00::/8) followed by a 40-bit
pseudo-random Global ID to produce the prefix [ULA]::/48, which is
then followed by a 16-bit Subnet ID then finally followed by a 64 bit

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

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

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

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

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

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

o etc.

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

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

include all 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).

Each OMNI link may be subdivided into SRT segments that often
correspond to different administrative domains or physical
partitions. OMNI nodes can use IPv6 Segment Routing [RFC8402] when
necessary to support efficient forwarding to destinations located in
other OMNI link segments. A full discussion of Segment Routing over
the OMNI link appears in [I-D.templin-6man-aero].

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.

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

For IPv4, GUA prefixes are assigned by IANA [IPV4-GUA] and/or an
associated regional assigned numbers authority such that the OMNI
domain can be interconnected to the global IPv4 Internet without
causing routing inconsistencies. An OMNI domain could instead use

private IPv4 prefixes (e.g., 10.0.0.0/8, etc.) [RFC3330], however
this would require IPv4 NAT if the domain were ever connected to the
global IPv4 Internet.

11. Node Identification

OMNI 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 12.2.13). One useful
identification value alternative is the Host Identity Tag (HIT) as
specified in [RFC7401], while Hierarchical HITs (HHITs)
[I-D.ietf-drip-rid] may provide an alternative more appropriate for
certain domains such as the Unmanned (Air) Traffic Management (UTM)
service for Unmanned Air Systems (UAS). Another alternative is the
Universally Unique IDentifier (UUID) [RFC4122] which can be self-
generated by a node without supporting infrastructure with very low
probability of collision.

When a 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 to a FHS Proxy/Server over (non-multihop)
protected-spectrum ANETs, an alternate form of node identification
(e.g., MAC address, serial number, airframe identification value,
VIN, etc.) may be sufficient. The Client can then include OMNI "Node
Identification" sub-options (see: Section 12.2.13) in IPv6 ND
messages should the need to transmit identification information over
the network arise.

12. Address Mapping - Unicast

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

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 10:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 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                          ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
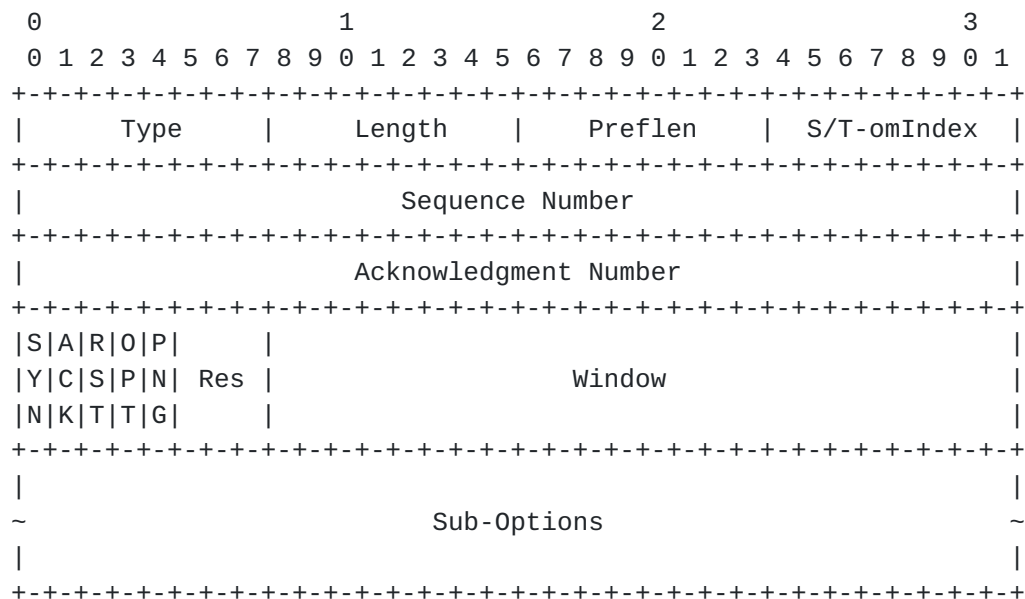
Figure 10: OMNI Option Format

In this format:

o Type is set to TBD2.

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

o Preflen is an 8 bit field that determines the length of prefix
   associated with an LLA. Values 0 through 128 specify a valid
   prefix length (all other values are invalid). For IPv6 ND
   messages sent from a 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.

o The remaining header fields before the Sub-Options begin 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 when TCP-like window synchronization is
   used, while the TCP (URG, PSH, FIN) flags are never 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 6.5.

o Sub-Options is a Variable-length field 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 the Sub-Options of 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 union of the information in the most recently received OMNI
options is therefore retained and aged/removed in conjunction with
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 after 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 11: Sub-Option Format

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

```
Sub-Option Name                Sub-Type
Pad1                               0
PadN                               1
Interface Attributes (Type 1)      2
Interface Attributes (Type 2)      3
Interface Attributes (Type 4)      4
MS-Register                        5
MS-Release                         6
Geo Coordinates                    7
DHCPv6 Message                     8
HIP Message                        9
PIM-SM Message                    10
Reassembly Limit                  11
Fragmentation Report              12
Node Identification               13
ICMPv6 Error                      14
Sub-Type Extension                30
```

Figure 12

Sub-Types 14-29 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.

During transmission, the OMNI interface codes Sub-Type and Sub-Length together in network byte order in 2 consecutive octets, where Sub-Option Data may be up to 2040 octets minus the length of the OMNI option header octets preceding the Sub-Options. This allows ample space for coding large objects (e.g., ASCII strings, domain names, protocol messages, security codes, etc.), while a single OMNI option is limited to 2040 octets the same as for any IPv6 ND option. 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.

During reception, the OMNI interface processes the OMNI option Sub-Options 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 total length, the OMNI interface accepts any sub-options
already processed and ignores the final sub-option. The interface
then processes any remaining OMNI options in the same fashion to the
end of the IPv6 ND message.

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

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

12.2.1. Pad1

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

Figure 13: Pad1

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

o Sub-Type is followed by 3 'x' bits, set to any value on
   transmission (typically all-zeros) and ignored on receipt. Pad1
   therefore consists of 1 octet with the most significant 5 bits set
   to 0, and with no Sub-Length or Sub-Option Data fields following.

12.2.2. 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 14: PadN

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

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

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

12.2.3. Interface Attributes (Types 1 through 3)

   Interface Attributes (Type 1) and (Type 2) were defined in
   [I-D.templin-6man-omni-interface] and have been moved to historic
   status. Their sub-option types (2 and 3) are reserved for future
   use.

   Interface Attributes (Type 3) was never defined; the number was
   skipped to bring (Type 4) into agreement with the corresponding sub-
   option Type value.

12.2.4. Interface Attributes (Type 4)

   The Interface Attributes (Type 4) sub-option provides L2 forwarding
   information for the multilink conceptual sending algorithm discussed
   in Section 14. The L2 information is used for selecting among
   potentially multiple candidate underlying interfaces that can be used
   to forward carrier packets to the neighbor based on factors such as
   traffic selectors and link quality. Interface Attributes (Type 4)
   further includes link-layer address information to be used for either
   OAL encapsulation or direct UDP/IP encapsulation (when OAL
   encapsulation can be avoided).

   Interface Attributes (Type 4) must be honored by all implementations.
   Throughout the remainder of this specification, when the term
   "Interface Attributes" appears without a "Type" designation the below
   format is indicated:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | S-Type=4|   Sub-length=N     |    omIndex    |   TS Format   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    omType    | Provider ID | Link  | Resvd | FMT |    SRT    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             LHS                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                                                               ~
   ~                  Link Layer Address (L2ADDR)                  ~
   ~                                                               ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                                                               ~
   ~                  RFC 6088 Format Traffic Selector             ~
   ~                                                               ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                 Figure 15: Interface Attributes (Type 4)

   o Sub-Type is set to 4. If multiple instances with different
      omIndex values appear in OMNI options of the same message all are
      processed. If multiple instances with the same omIndex value
      appear, the Traffic Selectors of all are processed while the
      remaining information is processed only for the first instance and
      ignored in all other instances.

   o Sub-Length is set to N that encodes the number of Sub-Option Data
      octets that follow. All fields beginning with omIndex up to and
      including TS Format are always present, while the 'A' and 'T'
      flags determine the remaining Sub-Option Data format.

   o Sub-Option Data contains an "Interface Attributes (Type 4)" 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 same message
         may include multiple Interface Attributes sub-options, with
         each distinct omIndex value pertaining to a different
         underlying interface. The OMNI option will often include an
         Interface Attributes sub-option with the same omIndex value
         that appears in the S/T-omIndex. In that case, the actual
         encapsulation address of the received IPv6 ND message should be
         compared with the L2ADDR encoded in the sub-option (see below);
         if the addresses are different the presence of a NAT is
         indicated.

* 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, the Traffic Selector includes IPv4 information. If it
  encodes the value 2, the Traffic Selector includes IPv6
  information. If it encodes the value 0, the Traffic Selector
  field is omitted.

* 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.

* The following address-related fields appear next in consecutive
  order:

  + FMT - a 3-bit "Forward/Mode/Type" code corresponding to the
    included Link Layer Address as follows:

    - When the most significant bit (i.e., "FMT-Forward") is
      clear, the LHS Proxy/Server must reassemble. When the
      bit is set, the LHS Proxy/Server must forward the
      fragments to the Client (while changing the OAL
      destination address) without reassembling.

    - When the next most significant bit (i.e., "FMT-Mode") is
      clear, L2ADDR is the address of the LHS Proxy/Server and
      the Client must be reached through the LHS Proxy/Server.
      When the bit is set, the Client can be reached on the
      open *NET where it may be located behind one or more NATs
      and L2ADDR is either the address of the LHS Proxy/Server
      (when FMT-Forward is set) or the native INET address of
      the Client itself (when FMT-Forward is clear).

    - 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. When SRT is set to 0 and LHS is non-zero, the
   prefix length is set to 128. SRT and LHS together provide
   guidance to the OMNI interface forwarding algorithm.
   Specifically, if SRT/LHS is located in the local OMNI link
   segment then the OMNI interface can encapsulate according to
   FMT/L2ADDR (following any necessary NAT traversal
   messaging); else, it must forward according to the SRT
   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 UDP Port Number appears in the
   first 2 octets and the IP address appears in the next 4
   octets for IPv4 or 16 octets for IPv6. The Port Number and
   IP address are recorded in network byte order, and in ones-
   compliment "obfuscated" form per [RFC4380]. The OMNI
   interface forwarding algorithm uses FMT/L2ADDR to determine
   the encapsulation address for forwarding when SRT/LHS is
   located in the local OMNI link segment.

* 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. Note that each
   Interface Attributes sub-option includes at most one IPv4 or
   IPv6 Traffic Selector block. 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 sub-options all
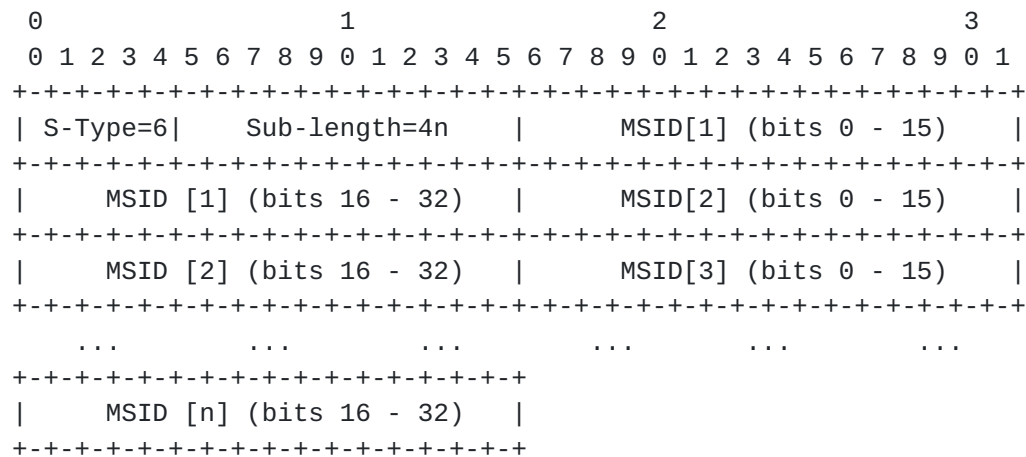   having the same omIndex in the following format:

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

12.2.5. MS-Register

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| S-Type=5|    Sub-length=4n    |     MSID[1] (bits 0 - 15)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    MSID [1] (bits 16 - 32)    |     MSID[2] (bits 0 - 15)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    MSID [2] (bits 16 - 32)    |     MSID[3] (bits 0 - 15)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      ...         ...         ...         ...        ...         ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    MSID [n] (bits 16 - 32)    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

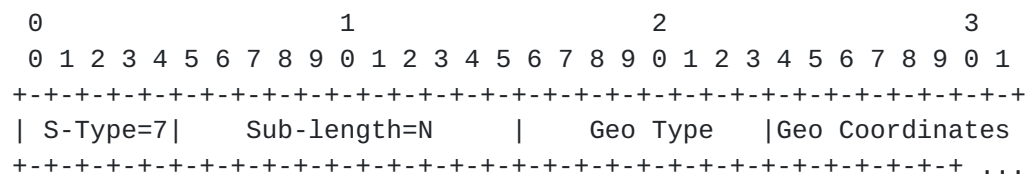   Figure 16: MS-Register Sub-option

   o Sub-Type is set to 5. If multiple instances appear in OMNI
      options of the same message all are processed. Only the first
      MAX_MSID values processed (whether in a single instance or
      multiple) are retained and all other MSIDs are ignored.

   o Sub-Length is set to 4n, with n representing the number of MSIDs
      included.

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

12.2.6. MS-Release

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | S-Type=6|    Sub-length=4n    |      MSID[1] (bits 0 - 15)    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      MSID [1] (bits 16 - 32)  |      MSID[2] (bits 0 - 15)    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      MSID [2] (bits 16 - 32)  |      MSID[3] (bits 0 - 15)    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       ...        ...        ...        ...       ...         ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      MSID [n] (bits 16 - 32)  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
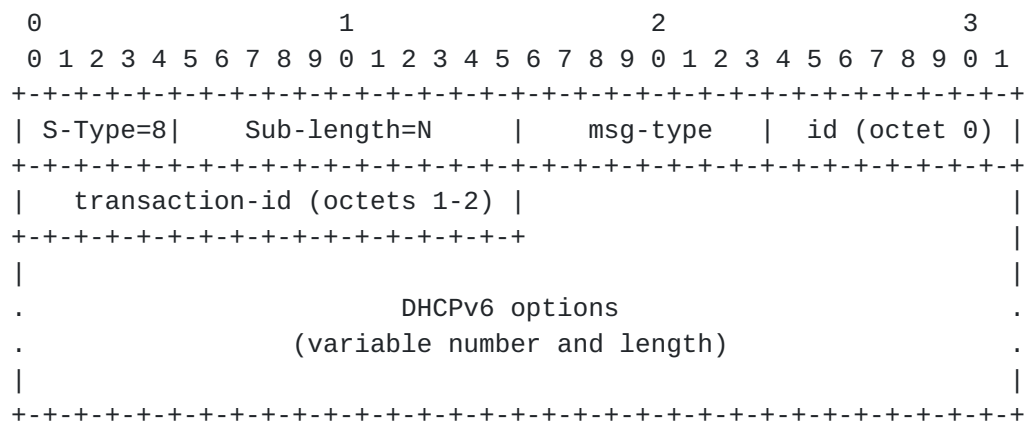
Figure 17: MS-Release Sub-option

o Sub-Type is set to 6. If multiple instances appear in OMNI
   options of the same message all are processed. Only the first
   MAX_MSID values processed (whether in a single instance or
   multiple) are retained and all other MSIDs are ignored.

o Sub-Length is set to 4n, with n representing the number of MSIDs
   included.

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

12.2.7. 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=7|    Sub-length=N     |    Geo Type   |Geo Coordinates
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ...
```

Figure 18: Geo Coordinates Sub-option

o Sub-Type is set to 7. 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 Sub-Option Data
   octets that follow.

o 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.

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

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

   The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) sub-option
   may be included in the OMNI options of Client RS messages and Proxy/
   Server RA messages. FHS Proxy/Servers that forward RS/RA messages
   between a Client and an LHS Proxy/Server also forward DHCPv6 Sub-
   Options unchanged. Note that DHCPv6 messages do not include a
   Checksum field, but integrity is protected by the IPv6 ND message
   Checksum.

```
      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     |   msg-type  |  id (octet 0) |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   transaction-id (octets 1-2) |                             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                             |
     |                                                             |
     .                      DHCPv6 options                         .
     .                 (variable number and length)               .
     |                                                             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
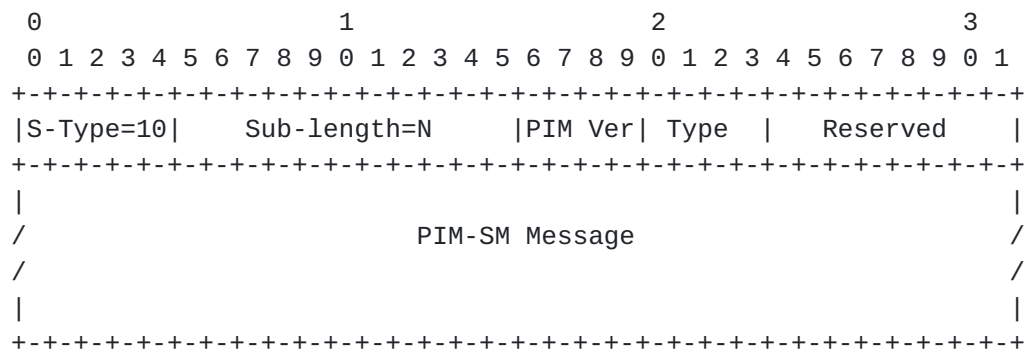
                   Figure 19: DHCPv6 Message Sub-option

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

   o Sub-Length is set to N 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.

12.2.9. **Host Identity Protocol (HIP) Message**

   The Host Identity Protocol (HIP) Message sub-option may be included
   in the OMNI options of IPv6 ND messages exchanged between Clients and
   FHS Proxy/Servers over an open Internetwork. FHS Proxy/Servers
   authenticate the HIP signatures of 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
   signatures before forwarding them to the Client.

   The HIP message sub-option should be included in any OMNI IPv6 ND
   message that traverses an open Internetwork, i.e., where link-layer
   authentication is not already assured by lower layers.

```
    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    |0| Packet Type |Version| RES.|1|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           Reserved          |            Controls            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |              Sender's Host Identity Tag (HIT)                |
   |                                                             |
   |                                                             |
   |                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |             Receiver's Host Identity Tag (HIT)              |
   |                                                             |
   |                                                             |
   |                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                             |
   /                      HIP Parameters                         /
   /                                                             /
   |                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
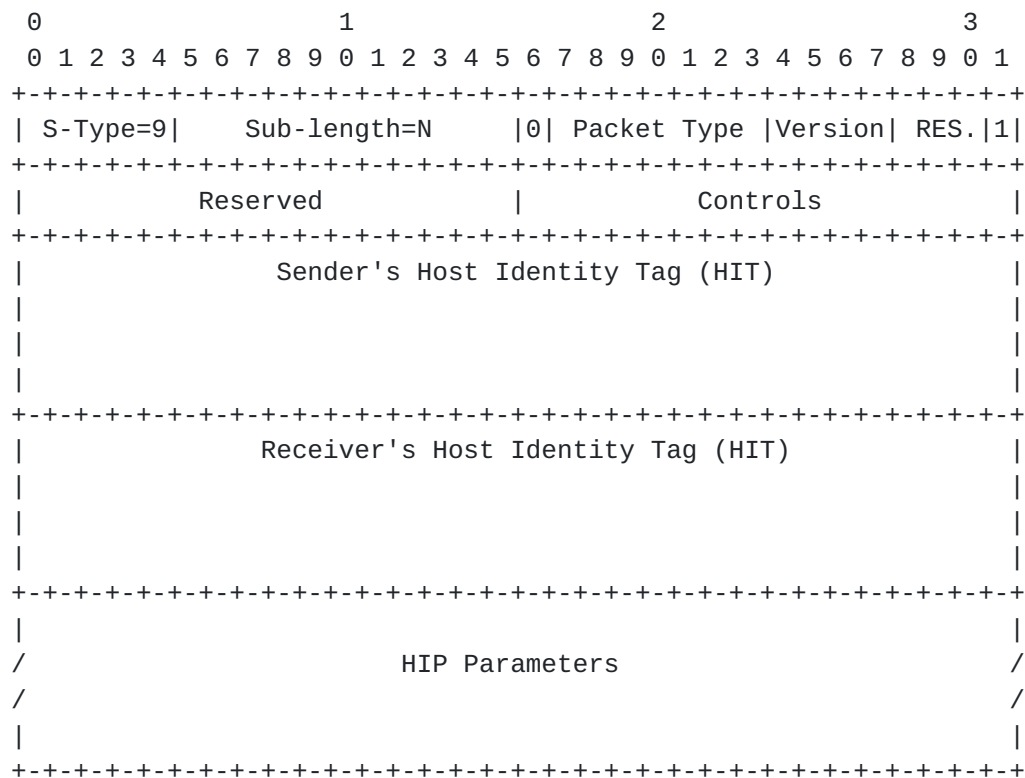
                   Figure 20: HIP Message Sub-option

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

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

o 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 header already
   includes a Checksum 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, etc.) can be used instead
of HITs in the authentication signature as long as the address can be
uniquely associated with the Sender/Receiver.

12.2.10. PIM-SM Message

The Protocol Independent Multicast - Sparse Mode (PIM-SM) Message
sub-option may be included in the OMNI options of IPv6 ND messages.
PIM-SM messages are formatted as specified in Section 4.9 of
[RFC7761], with the exception that the Checksum field is replaced by
a Reserved field (set to 0) since the IPv6 ND message header already
includes a Checksum. The PIM-SM message sub-option format is shown
in Figure 21:

```
     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    |PIM Ver| Type  |   Reserved    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    /                       PIM-SM Message                          /
    /                                                               /
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                  Figure 21: PIM-SM Message Option Format
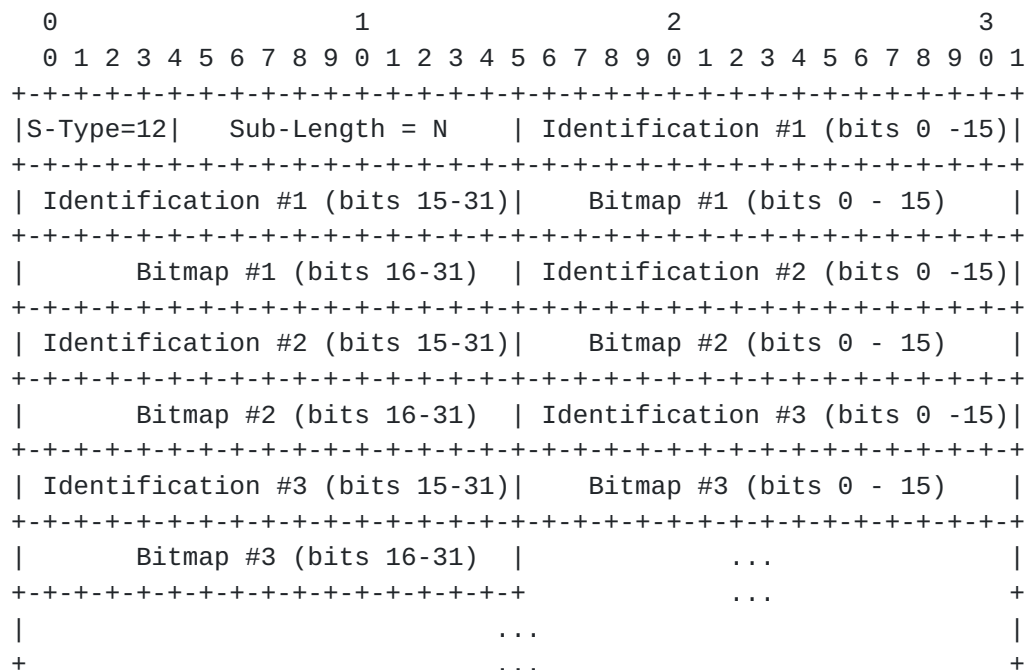
o Sub-Type is set to 10. If multiple instances appear in OMNI
   options of the same message all are processed.

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 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.)

12.2.11. 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=11|   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)               |
   +                                                               +
```
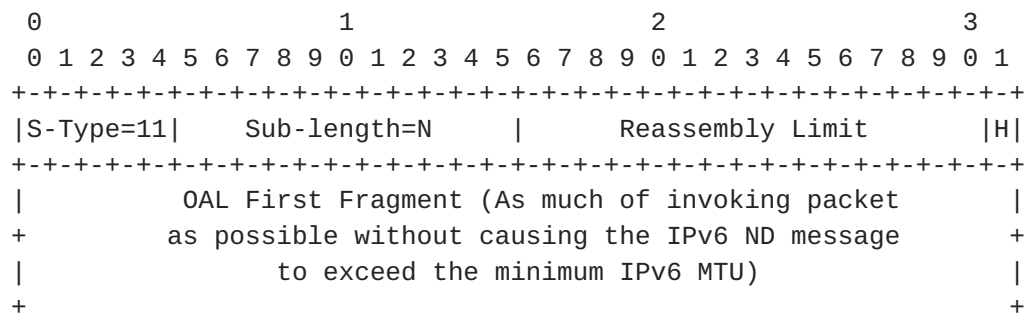
                    Figure 22: Reassembly Limit

   o Sub-Type is set to 11. 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.

12.2.12. Fragmentation Report

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

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |S-Type=12|   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 23: Fragmentation Report

   o Sub-Type is set to 12. 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
      Bitmp 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 would encode:

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

                              Figure 24

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

12.2.13. Node Identification

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |S-Type=13|    Sub-length=N    |    ID-Type    |              ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+              ~
      ~             Node Identification Value (N-1 octets)          ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                    Figure 25: Node Identification

   o Sub-Type is set to 13. If multiple instances appear in OMNI
      options of the same IPv6 ND message the first instance of a
      specific ID-Type is processed and all other instances of the same
      ID-Type are ignored. (Note therefore that it is possible for a
      single IPv6 ND message to convey multiple Node Identifications -
      each having a different ID-Type.)

o Sub-Length is set to N 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.

When a Node Identification Value is used for DHCPv6 messaging
purposes, it is encoded as a DHCP Unique IDentifier (DUID) using the
"DUID-EN for OMNI" format with enterprise number 45282 (see:
Section 25) as shown in Figure 26:

```
      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 26: DUID-EN for OMNI Format

In this format, the ID-Type and Node Identification Value fields are
coded exactly as in Figure 25 following the 6 octet DUID-EN header,
and the entire "DUID-EN for OMNI" is included in a DHCPv6 message per
[RFC8415].

12.2.14. 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=14|    Sub-length=N     |                              ~
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-                              ~
     ~                       RFC4443 Error Message Body             ~
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 27: ICMPv6 Error

o Sub-Type is set to 14. 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
   error messages are processed exactly per the standard, while
   ICMPv6 informational messages must not be included and are 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.

12.2.15. 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 28:

```
 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                    ~
~                                                              ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
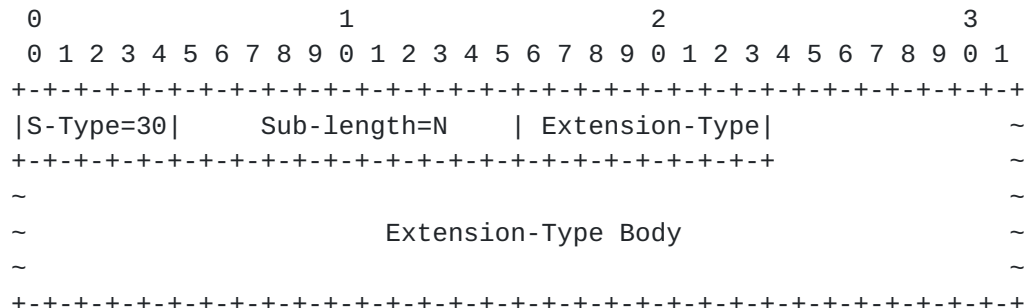
                    Figure 28: Sub-Type Extension

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

o Sub-Length is set to N 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.

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

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

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

12.2.15.1. RFC4380 UDP/IP Header 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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
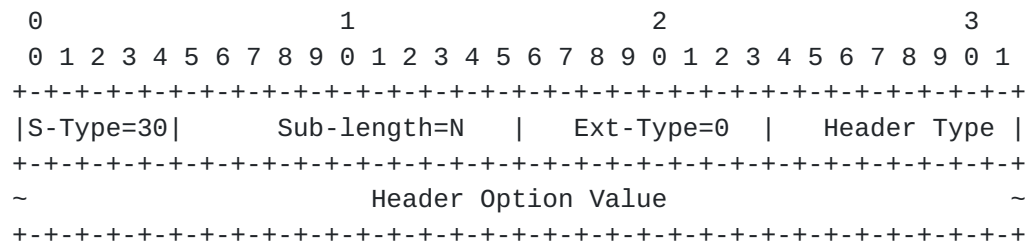
   Figure 29: RFC4380 UDP/IP Header Option (Extension-Type 0)

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.

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.

12.2.15.2. RFC6081 UDP/IP Trailer 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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
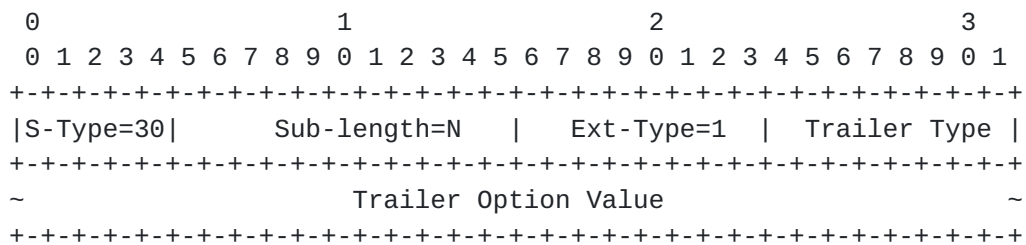
      Figure 30: RFC6081 UDP/IP Trailer Option (Extension-Type 1)

   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] while using the
L2 address of a Proxy/Server as the L2 address for all multicast
packets.

The Client uses Multicast Listener Discovery (MLDv2) [RFC3810] to
coordinate with Proxy/Servers, and *NET L2 elements use MLD snooping
[RFC4541].

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
with 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.

After an original IP packet enters the OMNI interface, one or more
outbound underlying interfaces are selected based on PBM traffic
attributes, and one or more neighbor underlying interfaces are
selected based on the receipt of Interface Attributes sub-options in
IPv6 ND messages (see: Section 12.2.4). Underlying interface
selection for the node's own local interfaces are based on traffic
selectors, cost, performance, message size, etc. Both node-local and
neighbor underlying interface traffic selectors may also be
configured to indicate replication for increased reliability at the
expense of packet duplication. The set of all Interface Attributes
received in IPv6 ND messages determines the multilink forwarding
profile for selecting the neighbor's underlying interfaces.

When the OMNI interface sends an original IP packet over a selected
outbound underlying interface, the OAL employs encapsulation and

fragmentation as discussed in Section 5, then performs *NET
encapsulation as determined by the L2 address information received in
Interface Attributes. The OAL also performs encapsulation when the
FHS 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 concurrently
in support of SBM. Each OMNI interface is distinguished by its
Anycast ULA (e.g., [ULA]:0002::, [ULA]:1000::, [ULA]:7345::, etc.).
The Client configures a separate OMNI interface for each link so that
multiple interfaces (e.g., omni0, omni1, omni2, etc.) are exposed to
the IPv6 layer. A different Anycast ULA is assigned to each
interface, and the Client injects the service prefixes for the OMNI
link instances into the EUN routing system.

Applications in EUNs can use Segment Routing to select the desired
OMNI interface based on SBM considerations. The Anycast ULA is
written into an original IP packet's IPv6 destination address, and
the actual destination (along with any additional intermediate hops)
is written into the Segment Routing Header. Standard IP routing
directs the packet to the Client's mobile router entity, and the
Anycast ULA identifies the OMNI interface to be used for transmission
to the next hop. When the Client receives the packet, it replaces
the IPv6 destination address with the next hop found in the routing
header and transmits the message over the OMNI interface identified
by the Anycast ULA.

Multiple distinct OMNI links can therefore be used to support fault
tolerance, load balancing, reliability, etc. The architectural model
is similar to Layer 2 Virtual Local Area Networks (VLANs).

14.2. 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 destined to an MNP address to the

Proxy/Server as its default router. The Proxy/Server therefore must
drop any packets originating from the Client with a destination
address that matches the Client's registered MNP. The Proxy/Server
therefore drops any original IP packets or carrier packets with both
source and destination address corresponding to the same Client's MNP
(I.e., whether LLA, ULA or GUA).

15. Router Discovery and Prefix Registration

Clients interface with the MS by sending RS messages with OMNI
options under the assumption that one or more FHS Proxy/Servers on
the *NET will process the message and respond. The RS message is
received by one or more FHS Proxy/Servers, which may in turn forward
a proxied copy of the RS to any "LHS" Proxy/Servers for the Client's
other underlying interfaces. (Note that each Client underlying
interface has one or more FHS Proxy/Server, but that the Proxy/Server
for underling interface "B" may appear as an LHS Proxy/Server taken
from the perspective of underlying interface "A".) The Client then
configures default routes for the OMNI interface based on any RA
message responses.

For each underlying interface, the Client sends an RS message with an
OMNI option to coordinate with FHS/LHS Proxy/Servers identified by
MSID values. Example MSID discovery methods are given in [RFC5214]
and include data link login parameters, name service lookups, static
configuration, a static "hosts" file, etc. The RS message may
include either a multicast or unicast LLA destination address. When
the LLA destination is multicast, all FHS Proxy/Servers process the
RS. When the LLA destination is unicast, only the Proxy/Server that
configures that LLA processes the RS. When an destination is that of
an LHS Proxy/Server, the FHS Proxy/Server acts as a "proxy" to
facilitate the Client-LHS Proxy/Server RS/RA exchange without
creating NCE state as discussed below.

When an FHS Proxy/Server receives an RS with destination set to its
own ADM-LLA, it returns an RA while also forwarding a proxied version
of the RS to any LHS Proxy/Servers corresponding to non-zero OMNI
option MS-Register/Release MSIDs. The FHS Proxy/Server sets the RS
source address to its own ADM-LLA and sets the destination address to
the ADM-LLA of the LHS Proxy/Server. The FHS Proxy/Server does not
include any window synchronization parameters since the RS message
did not originate from the Client. When the LHS Proxy/Server
receives the RS message, it updates the Client's NCE ReachableTime
state but does not update the reachability state of the underlying
interface(s) for which it acts as an FHS Proxy/Server for this same
Client since the RS message did not test the reachability of the
underlying interface(s). The LHS Proxy/Server then returns an RA
message to the FHS Proxy/Server, which returns a proxyed (singular or

aggregate) RA to the Client. Note that while this multi-RS/RA
approach tests only the underlying interface between the Client and
FHS Proxy/Server, it may be useful for updating reachability
information with multiple Proxy/Servers in a single message exchange.

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, Interface
Attributes appropriate for underlying interfaces, MS-Register/Release
sub-options containing MSID values, Reassembly Limits, an
authentication sub-option and with any other necessary OMNI sub-
options. The OMNI interface then sets the S/T-omIndex field to the
index of the underlying interface over which the RS message is sent.

The OMNI interface then sends the RS over the underlying interface
using OAL encapsulation and fragmentation if necessary. If the
Client uses OAL encapsulation for RS messages sent over an INET
interface, the entire RS message must fit within a single carrier
packet (i.e., an atomic fragment) so that the first-hop Proxy/Server
can verify the authentication signature without having to reassemble.
The OMNI interface selects an Identification value (see:
Section 6.5), 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 site-scoped All-Routers
multicast (ff05::2) 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 forwarding proxied versions to LHS Proxy/Servers
named in the Register/Release list. When each Proxy/Server processes

the RS OMNI information, it first validates the prefix registration
information then injects/withdraws the MNP in the MS and caches/
discards the new Preflen, MNP and Interface Attributes. The Proxy/
Server then returns an RA message with an OMNI option per Section 12.

LHS Proxy/Servers return RAs to the FHS Proxy/server, and the FHS
Proxy/Server forwards them 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 source 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, Interface Attributes, MS-Register/Release 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:

o PIOs with (A; L=0) that include MSPs for the link [RFC8028].

o RIOs [RFC4191] with more-specific routes.

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

The FHS Proxy/Server prepares the RA using OAL encapsulation/
fragmentation with an Identification value selected per Section 6.5,
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
initial RA messages to the Client and MAY later send additional
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.

The FHS Proxy/Server can combine the information from multiple LHS
Proxy/Servers by sending one or more "aggregate" RAs to the Client in
order conserve *NET bandwidth. Each aggregate RA includes an OMNI
option with MS-Register/Release sub-options with the MSIDs of all LHS
Proxy/Servers represented by the aggregate. Each such aggregate RA
message must consistently represent the combined information
advertised by all represented LHS Proxy/Servers. Note that since the
FHS Proxy/Server uses its own ADM-LLA as the RA source address, the
Client determines the addresses of the represented LHS Proxy/Servers
by examining the MS-Register/Release OMNI sub-options. Note also
that the FHS Proxy/Server must return any LHS Proxy/Server RA
messages that set window synchronization flags directly to the
Client, i.e., and without including them in an aggregate.

When the Client receives the RA message, it creates an OMNI interface
NCE for each MSID that has confirmed MNP registration via the L2
address of the FHS Proxy/Server. If the Client connects to multiple
*NETs, it records the additional L2 FHS Proxy/Server addresses in
each MSID NCE (i.e., as multilink neighbors). 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:

o When an underlying interface transitions to UP, the Client sends
   an RS over the underlying interface with an OMNI option. The OMNI
   option contains at least one Interface Attribute sub-option with
   values specific to this underlying interface, and may contain
   additional Interface Attributes specific to other underlying
   interfaces. The option also includes any MS-Register/Release sub-
   options.

o When an underlying interface transitions to DOWN, the Client sends
   an RS or unsolicited NA message over any UP underlying interface
   with an OMNI option containing an Interface Attribute sub-option
   for the DOWN underlying interface with Link set to '0'. The
   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 SYN
   flag in the OMNI header to trigger a reliable solicited NA reply.

o When the Router Lifetime for a FHS Proxy/Server nears expiration,
   the Client sends an RS over the underlying interface to receive a
   fresh RA. If no RA messages are received (i.e., after retrying),
   the Client marks the underlying interface as DOWN.

o When a Client wishes to release from one or more current MSIDs, it
   sends an RS or unsolicited NA message over any UP underlying
   interfaces with an OMNI option with a Release MSID. Each MSID
   then withdraws the MNP from the routing/mapping system and informs
   the Proxy/Server that the release was successful.

o 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: 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.

Note: When a FHS Proxy/Server forwards a Client's RS message to an
LHS Proxy/Server using UDP/IP encapsulation, it must use a distinct
UDP source port number for each Client. This allows the LHS Proxy/
Server to distinguish different Clients behind the same FHS Proxy/
Server at the link-layer, whereas the link-layer addresses would
otherwise be indistinguishable.

Note: When a Proxy/Server returns an RA to an INET Client, it
includes an OMNI option with an Interface Attributes sub-option with
omIndex set to 0 and with SRT, FMT, LHS and L2ADDR information for

its INET interface. This provides the Client with SRT segment prefix
context regarding the local OMNI link segment.

15.1. Window Synchronization

In environments where Identification window synchronization is
necessary, the RS/RA exchanges discussed above observe the procedures
specified in Section 6.5. In the asymmetric case, the initial RS/RA
exchange establishes only the Client's send window and Proxy/Server's
receive window such that a corresponding NS/NA exchange would be
needed in the reverse direction. In the symmetric case, the Client
returns an explicit/implicit acknowledgement response to the RA to
symmetrically establish the send/receive windows of both parties.

The initial RS/RA exchange between a Client and FHS Proxy/Server over
a first underlying interface must invoke window synchronization when
necessary, while subsequent RS/RA exchanges with the same Proxy/
Server performed over additional underlying interfaces within
ReachableTime and with in-window Identification values need not also
invoke 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 sends an RS SYN to a multicast destination address, it
may receive RA SYN/ACKs from multiple FHS Proxy/Servers - each with
their own synchronization parameters. The resulting "multi-three-
way" handshake would require the Client to establish separate NCE
SND/RCV state and return explicit/implicit acknowledgements for each
responding FHS Proxy/Server.

When a Client needs to perform window synchronization with an LHS
Proxy/Server via an FHS Proxy/Server, it sets the RS SYN source
address to its own MNP-LLA and destination address to the ADM-LLA of
the LHS 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, then forwards the resulting carrier
packets to the FHS Proxy/Server. The FHS Proxy/Server authenticates
the message, then re-encapsulates it with its own ADM-ULA as the
source and the ADM-ULA of the LHS Proxy/Server as the target.

The FHS Proxy/Server then securely forwards the carrier packets to
the LHS Proxy/Server, which updates its window synchronization
information for the Client and returns a secured unicast RA message
with source set to its own ADM-LLA and destination set to the
Client's MNP-LLA. The LHS 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 securely forwards the

carrier packets to the FHS Proxy/Server. The FHS Proxy/Server then
re-encapsulates the message using its own ADM-ULA as the source, the
MNP-ULA of the Client as the destination, and with an authentication
signature included if necessary. The FHS Proxy/Server then forwards
the message to the Client which updates its window synchronization
information for the LHS Proxy/Server (i.e., and not the FHS Proxy/
Server which is simply acting as a proxy).

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

On some *NETs, a Client may be located multiple IP hops away from an
FHS 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.). These *NETs
could be either IPv6-enabled or IPv4-only, while IPv4-only *NETs
could be either multicast-capable or unicast-only (note that for
IPv4-only *NETs the following procedures apply for both single-hop
and multihop cases).

A Client located potentially multiple *NET hops away from an FHS
Proxy/Server prepares an RS message with source address set to its
MNP-LLA (or to the unspecified address (::) if it does not yet have
an MNP-LLA), and with destination set to link-scoped All-Routers
multicast the same as discussed above. The OMNI interface then
employs OAL encapsulation and fragmentation, and sets the OAL source
address to the ULA corresponding to the RS source (or to a Temporary
ULA if the RS source was the unspecified address (::)) and sets the
OAL destination to site-scoped All-Routers multicast (ff05::2). For
IPv6-enabled *NETs, the Client then encapsulates the message in UDP/
IPv6 headers with source address set to the underlying interface
address (or to the ULA that would be used for OAL encapsulation if
the underlying interface does not yet have an address) and sets the
destination to either a unicast or anycast Proxy/Server IPv6 address.
For IPv4-only *NETs, the Client instead encapsulates the RS message
in UDP/IPv4 headers with source address set to the IPv4 address of
the underlying interface and with destination address set to either a
unicast Proxy/Server IPv4 address [RFC5214] or an IPv4 anycast
address reserved for OMNI. The Client then sends the encapsulated RS
message via the *NET interface, where it will be forwarded by zero or
more intermediate *NET hops.

When an intermediate *NET hop that participates in the routing
protocol receives the encapsulated RS, it forwards the message
according to its routing tables (note that an intermediate node could
be a fixed infrastructure element or another Client). This process
repeats iteratively until the RS message is received by a penultimate

*NET hop within single-hop communications range of an FHS Proxy/
Server, which forwards the message to the Proxy/Server.

When the FHS Proxy/Server receives the message, it decapsulates the
RS (while performing OAL reassembly, if necessary) and coordinates
with the MS the same as for an ordinary link-local RS, since the
network layer Hop Limit will not have been decremented by the
multihop forwarding process. The FHS Proxy/Server then prepares an
RA message with source address set to its own ADM-LLA and destination
address set to the LLA of the original Client. The FHS 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 RA source. The FHS Proxy/Server then encapsulates the message
in UDP/IPv4 or UDP/IPv6 headers with source address set to its own
address and with destination set to the encapsulation source of the
RS.

The FHS 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 an FHS Proxy/Server on the same physical link.

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

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

Note: An IPv4 anycast address for OMNI in IPv4 networks could be part
of a new IPv4 /24 prefix allocation, but this may be difficult to
obtain given IPv4 address exhaustion. OMNI therefore proposes to re-
claim the prefix 192.88.99.0 [RFC7526] for this purpose (see IANA
considerations).

15.3. MS-Register and MS-Release List Processing

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

   When a Client sends an RS message with an OMNI option via an
   underlying interface to an FHS Proxy/Server, the Client can
   optionally convey information for any LHS Proxy/Servers. Initially,
   the Client will have no associated Proxy/Servers and should therefore
   send its initial RS messages to the link-scoped All-Routers multicast
   address. An FHS Proxy/Server will then return an RA message with
   source address set to its own ADM-LLA.

   As the Client activates additional underlying interfaces, it sends RS
   messages to receive RA messages from FHS Proxy/Servers specific to
   those interfaces. Thereafter, it can optionally send RS messages to
   FHS Proxy/Servers over a first underlying interface that include an
   MS-Register sub-option with MSIDs for LHS Proxy/Servers for other
   underlying interfaces. The Client will thus eventually begin to
   learn and manage its currently active set of Proxy/Servers, and can
   register with new Proxy/Servers or release from former Proxy/Servers
   with each successive RS/RA exchange. As the Client's Proxy/Server
   constituency grows, it alone is responsible for including or omitting
   MSIDs in the MS-Register/Release lists it sends in RS messages. The
   inclusion or omission of MSIDs determines the Client's interface to
   the MS and defines the manner in which Proxy/Servers will respond.
   The only limiting factor is that the Client should include no more
   than MAX_MSID values in each list per each IPv6 ND message, and
   should avoid duplication of entries in each list unless it wants to
   increase likelihood of control message delivery.

   When an FHS Proxy/Server receives an RS message sent by a Client with
   an OMNI option, the option will contain zero or more MS-Register and
   MS-Release sub-options containing MSIDs for LHS Proxy/Servers. After
   processing the OMNI option, the FHS Proxy/Server will have a list of
   zero or more MS-Register MSIDs and a list of zero or more of MS-
   Release MSIDs. The FHS Proxy/Server then processes the lists as
   follows:

   o For each list, retain the first MAX_MSID values in the list and
      discard any additional MSIDs (i.e., even if there are duplicates
      within a list).

   o Next, for each MSID in the MS-Register list, remove all matching
      MSIDs from the MS-Release list.

o Next, proceed as follows:

    * If the FHS Proxy/Server's own MSID appears in the MS-Register
       list, send an RA message directly back to the Client and send a
       proxy copy of the RS message to each additional LHS Proxy/
       Server MSID in the MS-Register list with the MS-Register/
       Release lists omitted. Then, send an unsolicited NA (uNA)
       message to each LHS Proxy/Server MSID in the MS-Release list
       with the MS-Register/Release lists omitted and with an OMNI
       option with S/T-omIndex set to 0.

    * Otherwise, send a proxy copy of the RS message to each
       additional LHS Proxy/Server MSID in the MS-Register list with
       the MS-Register list omitted. For the first LHS Proxy/Server
       MSID, include the original MS-Release list; for all other
       MSIDs, omit the MS-Release list.

Each proxy copy of the RS message will include an OMNI option and OAL
encapsulation header with the ADM-ULA of the FHS Proxy/Server as the
source and the ADM-ULA of the LHS Proxy/Server as the destination.
When the LHS Proxy/Server receives the proxy RS message, if the
message includes an MS-Release list the LHS Proxy/Server sends a uNA
message to each additional MSID in the Release list with an OMNI
option with S/T-omIndex set to 0. The LHS Proxy/Server then sends an
RA message back to the FHS Proxy/Server with an OAL header with
source and destination addresses reversed, and with RA destination
set to the MNP-LLA of the Client. When the FHS Proxy/Server receives
this RA message, it sends a proxy copy of the RA to the Client.

Each uNA message (whether sent by an FHS or LHS Proxy/Server) will
include an OMNI option and an OAL header with the ADM-ULA of the uNA
source Proxy/Server as the source and the ADM-ULA of uNA target
Proxy/Server as the destination. The uNA informs the target Proxy/
Server that its previous relationship with the Client has been
released and that the source of the uNA message is now registered.
The uNA target Proxy/Server must then note that the subject Client of
the uNA message is now "departed", and forward any subsequent packets
destined to the Client to the uNA source Proxy/Server.

Note: It is not an error for the MS-Register/Release lists to include
duplicate entries. If duplicates occur within a list, the FHS Proxy/
Server will generate multiple proxy RS and/or uNA messages - one for
each copy of the duplicate entries.

15.4. **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 an FHS 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 FHS 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 FHS 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 FHS Proxy/Server uses the
   DHCPv6 Solicit message contained in the OMNI option. The FHS 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 FHS 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 FHS 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 FHS Proxy/Server receives the DHCPv6 Reply, it adds routes
   to the routing system and creates MNP-LLAs based on the delegated
   MNPs. The FHS 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 FHS Proxy/Server includes one of the (newly-created) MNP-LLAs as
the RA destination address and sets the OMNI option Preflen
accordingly; otherwise, the FHS Proxy/Server includes the RS source
address as the RA destination address. The FHS 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. 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.

Note: After a Client performs a DHCPv6-based prefix registration
exchange with a first Proxy/Server, it would need to repeat the
exchange with each of its additional Proxy/Servers. In that case,
the Client supplies the MNP delegation information received from the
first Proxy/Server when it engages the additional Proxy/Servers.

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 returns an RA
with a non-zero Router Lifetime 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 LHS
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 FHS Proxy/Servers for
which there are currently active Clients on the *NET. If an LHS
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) LHS 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 returns an RA with its ADM-LLA as the
source, the MNP-LLA as the destination and with an OMNI option
included. 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 C. 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 RA messages that do not include an OMNI option, while
OMNI interfaces configured over IPv4-only underlying interfaces do
not receive any (IPv6) RA messages at all (although they may receive
IPv4 RA messages [RFC1256]). Client OMNI interfaces that receive RA
messages without an OMNI option configure addresses, on-link
prefixes, etc. on the underlying interface that received the RA
according to standard IPv6 ND and address resolution conventions
[RFC4861] [RFC4862]. 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 that
connect to an open Internetwork can apply security services such as
VPNs to connect to an FHS 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 use UDP/IP
encapsulation while including authentication signatures in IPv6 ND
messages.

OMNI interfaces use UDP service port number 8060 (see: Section 25.11
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 encapsulates the original IP packet or OAL packet
immediately following the UDP header, with the IP protocol version
identified by the first four bits. (When the first four bits include
a value other than 4 or 6, the UDP message body is interpreted
according to the OCH-0, OCH-1 or other header formats as discussed in
previous sections.) The OMNI interface sets the UDP length to the
exact length of the encapsulated IP or OAL packet, i.e., and must not
set a larger value to imply surplus space following the packet.

Since the OAL includes an integrity check over the OAL packet, OAL
sources selectively disable UDP checksums for OAL packets that do not
require ORH and/or UDP/IP address integrity, but enable UDP checksums
for others including non-OAL packets, IPv6 ND messages used to
establish link-layer addresses, etc. If the OAL source discovers
that packets with UDP checksums disabled are being dropped in the
path it should enable UDP checksums in future packets. Further
considerations for UDP encapsulation checksums are found in
[RFC6935][RFC6936].

For 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 HIP authentication is used, the IPv6 ND message source should
include an OMNI option with a HIP message containing a valid
authentication signature. When the source prepares the HIP message,
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 use ordinary IPv6 addresses instead of (H)HITs as long as the
Sender and Receiver have some way to associate the addresses with the
neighbor (e.g., via a node identifier embedded in the address).

Before calculating the HIP signature, the source sets both the ICMPv6
Checksum field and HIP signature fields to 0. The source then
calculates the HIP authentication signature over the full length of
the IPv6 ND message beginning with the ICMPv6 message header and
extending over all included IPv6 ND message options including the

OMNI option itself. The source next writes the authentication
signature into the HIP signature field, then calculates the ICMPv6
message checksum and writes the value into the ICMPv6 Checksum field.

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 and
mobility management (see: [I-D.templin-6man-aero]). These control
plane messages must be authenticated while data plane messages are
delivered the same as for ordinary best-effort traffic with source
address and/or Identification window-based data origin verification.
Data plane communications via OMNI interfaces that connect over open
Internetworks without an explicit VPN should therefore employ
transport- or higher-layer security to ensure integrity and/or
confidentiality.

Client OMNI interfaces configured over open Internetworks are often
located behind NATs. The OMNI interface accommodates NAT traversal
using UDP/IP encapsulation and the mechanisms discussed in
[I-D.templin-6man-aero]. To support NAT determination, Proxy/Servers
include an Origin Indication sub-option in RA messages sent in
response to RS messages received from a Client via UDP/IP
encapsulation.

Note: Following the initial IPv6 ND message exchange, OMNI interfaces
configured over open Internetworks 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: The [RFC4380] HMAC and/or HIP message [RFC7401] authentication
sub-options appear in the OMNI option, which may occur anywhere
within the IPv6 ND message body. When a node that inserts an
authentication sub-option generates the authentication signature, it
calculates the signature over the entire length of the IPv6 ND
message but with the sub-option authentication field itself set to 0.
The node then writes the resulting signature into the authentication
field then continues to prepare the message for transmission. For
this reason, if an IPv6 ND message includes multiple authentication
sub-options, the first sub-option is consulted and any additional
sub-options are ignored.

Note: OMNI interfaces on open Internetworks should employ the
Identification window synchronization mechanisms specified in
Section 6.5 in order to reject spurious carrier packets that might
otherwise clutter the reassembly cache. This is especially important
in environments where carrier packet spoofing 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 15.4 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 present limits for the durations of
   ongoing 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 since
   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 and with destination set to either a unicast or
   anycast ADM-ULA. The Client then sends the message as specified in
   Section 15.2.

   When an FHS 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 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. Note that the HIP security protocols for
establishing secure neighbor relationships are based on (H)HITs.
IPv6 ND messages that use Temporary ULAs instead use the HMAC
authentication service specified in [RFC4380].

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 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 that return error messages can include them 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. "IEEE 802 Numbers" Registry

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

25.2. "IPv6 Neighbor Discovery Option Formats" Registry

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

25.3. "Ethernet Numbers" Registry

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

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

            Figure 31: IANA Unicast 48-bit MAC Addresses

25.4. "ICMPv6 Code Fields: Type 2 - Packet Too Big" Registry

The IANA is instructed to assign two new Code values in the "ICMPv6
Code Fields: Type 2 - Packet Too Big" registry (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 32: ICMPv6 Code Fields: Type 2 - Packet Too Big Values

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

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

   The OMNI option defines a 5-bit Sub-Type field, for which IANA is
   instructed to create and maintain a new registry entitled "OMNI
   Option Sub-Type Values". Initial values are given below
   (registration procedure is RFC required):

```
     Value     Sub-Type name                Reference
     -----     -------------                ----------
     0         Pad1                         [RFCXXXX]
     1         PadN                         [RFCXXXX]
     2         Interface Attributes (Type 1)  [RFCXXXX]
     3         Interface Attributes (Type 2)  [RFCXXXX]
     4         Interface Attributes (Type 4)  [RFCXXXX]
     5         MS-Register                  [RFCXXXX]
     6         MS-Release                   [RFCXXXX]
     7         Geo Coordinates              [RFCXXXX]
     8         DHCPv6 Message               [RFCXXXX]
     9         HIP Message                  [RFCXXXX]
     11        PIM-SM Message               [RFCXXXX]
     11        Reassembly Limit             [RFCXXXX]
     12        Fragmentation Report         [RFCXXXX]
     13        Node Identification          [RFCXXXX]
     14        ICMPv6 Error                 [RFCXXXX]
     15-29     Unassigned
     30        Sub-Type Extension           [RFCXXXX]
     31        Reserved by IANA             [RFCXXXX]
```

                   Figure 33: OMNI Option Sub-Type Values

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

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

```
   Value    Sub-Type name              Reference
   -----    -------------              ----------
   0        NULL                       [RFCXXXX]
   255      Reserved by IANA           [RFCXXXX]
```

                   Figure 34: OMNI Geo Coordinates Type

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

   The OMNI Node Identification sub-option (see: Section 12.2.13)
   contains an 8-bit ID-Type field, for which IANA is instructed to
   create and maintain a new registry entitled "OMNI Node Identification
   ID-Type Values". Initial values are given below (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 35: OMNI Node Identification ID-Type Values

25.8. "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 36: OMNI Option Sub-Type Extension Values

25.9. "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 37: OMNI RFC4380 UDP/IP Header Option

25.10. "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 38: OMNI RFC6081 Trailer Option

25.11. 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-purpose the prefix 192.88.99.0 which has
been set aside from its former use by [RFC7526] as an IPv4 OMNI
interface anycast address.

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.
Control plane messages are then constrained to travel only over the
secured spanning tree paths and are therefore protected from attack
or eavesdropping. Since data plane messages can travel over route
optimized paths that do not strictly follow the spanning tree,
however, end-to-end transport- or higher-layer security services are
still required. Additionally, the OAL Identification value provides
a first level of data origin authentication that mitigates off-path
spoofing.

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 6.9. 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:

o [RFC1191]

o [RFC4443]

o [RFC8201]

o [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 as
early as Y2K, with insights from colleagues including Brian
Carpenter, Ralph Droms, Christian Huitema, Thomas Narten, Dave
Thaler, Joe Touch, and many others who deserve recognition.

Early observations on IP fragmentation performance implications were
noted in the 1986 Digital Equipment Corporation (DEC) "qe reset"
investigation, where fragment bursts from NFS UDP traffic triggered
hardware resets resulting in communication failures. Jeff Chase,

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.

[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>.

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

   [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group
              MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000,
              <https://www.rfc-editor.org/info/rfc2863>.

   [RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery",
              RFC 2923, DOI 10.17487/RFC2923, September 2000,
              <https://www.rfc-editor.org/info/rfc2923>.

   [RFC2983] Black, D., "Differentiated Services and Tunnels",
              RFC 2983, DOI 10.17487/RFC2983, October 2000,
              <https://www.rfc-editor.org/info/rfc2983>.

   [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition
              of Explicit Congestion Notification (ECN) to IP",
              RFC 3168, DOI 10.17487/RFC3168, September 2001,
              <https://www.rfc-editor.org/info/rfc3168>.

   [RFC3330] IANA, "Special-Use IPv4 Addresses", RFC 3330,
              DOI 10.17487/RFC3330, September 2002,
              <https://www.rfc-editor.org/info/rfc3330>.

   [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>.

   [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 Checksum
   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. VDL Mode 2 Considerations

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

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

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

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

Appendix C. 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 TBD3 (see: Section 25). For non-Ethernet
   statically-addressed *NETs, MSADDR is reserved per the assigned
   numbers authority for the *NET addressing space. For still other
   *NETs, MSADDR may be dynamically discovered through other means,
   e.g., L2 beacons.

   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 D. Change Log

   << RFC Editor - remove prior to publication >>

   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:

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

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

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

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

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

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

o 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:

o Change to S/T-omIndex definition.

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

o Changed reference citations to "draft-templin-6man-aero".

o Included introductory description of the "6M's".

o Included new OMNI sub-option for PIM-SM.

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

o Added citation of RFC8726.

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

o Updated IANA registration policies for OMNI registries.

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

o Changed intended document status to Informational, and removed documents from "updates" category.

o Updated implementation status.

o Minor edits to HIP message specifications.

o Clarified OAL and *NET IP header field settings during encapsulation and re-encapsulation.

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

o 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