Automatic Extended Route Optimization (AERO)
draft-templin-intarea-6706bis-97

Abstract

This document specifies an Automatic Extended Route Optimization
(AERO) service for IP internetworking over Overlay Multilink Network
(OMNI) interfaces. AERO/OMNI use an IPv6 link-local address format
that supports operation of the IPv6 Neighbor Discovery (ND) protocol
and links ND to IP forwarding. Prefix delegation/registration
services are employed for network admission and to manage the routing
system. Multilink operation, mobility management, multicast, quality
of service (QoS) signaling and route optimization are naturally
supported through dynamic neighbor cache updates. AERO is a widely-
applicable mobile internetworking service especially well-suited to
aviation services, intelligent transportation systems, mobile Virtual
Private Networks (VPNs) and many other applications.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   Automatic Extended Route Optimization (AERO) fulfills the
   requirements of Distributed Mobility Management (DMM) [RFC7333] and
   route optimization [RFC5522] for aeronautical networking and other
   network mobility use cases such as intelligent transportation
   systems. AERO is an internetworking and mobility management service
   that employs the Overlay Multilink Network Interface (OMNI)
   [I-D.templin-6man-omni-interface] Non-Broadcast, Multiple Access
   (NBMA) virtual link model. The OMNI link is a virtual overlay
   configured over one or more underlying Internetworks, and nodes on
   the link can exchange original IP packets as single-hop neighbors via
   encapsulation and fragmentation. The OMNI Adaptation Layer (OAL)
   supports multilink operation for increased reliability, bandwidth
   optimization and traffic path selection while performing
   fragmentation and reassembly to accommodate Maximum Transmission Unit
   (MTU) diversity.

   The AERO service comprises Clients, Proxy/Servers and Relays that are
   seen as OMNI link neighbors as well as Bridges that interconnect OMNI
   link segments through OAL forwarding at a layer below IP. Each
   node's OMNI interface uses an IPv6 link-local address format that
   supports operation of the IPv6 Neighbor Discovery (ND) protocol
   [RFC4861] and links ND to IP forwarding. A node's OMNI interface can
   be configured over multiple underlying interfaces, and therefore
   appears as a single interface with multiple link-layer addresses.
   Each link-layer address is subject to change due to mobility and/or
   QoS fluctuations, and link-layer address changes are signaled by ND
   messaging the same as for any IPv6 link.

   AERO provides a cloud-based service where mobile node Clients may use
   any Proxy/Server acting as a Mobility Anchor Point (MAP) and fixed
   nodes may use any Relay on the link for efficient communications.
   Fixed nodes forward original IP packets destined to other AERO nodes
   via the nearest Relay, which forwards them through the cloud. A
   mobile node's initial packets are forwarded through the Proxy/Server,
   and direct routing is supported through automatic extended route
   optimization while packets are flowing. Both unicast and multicast
   communications are supported, and mobile nodes may efficiently move
   between locations while maintaining continuous communications with
   correspondents and without changing their IP Address.

   AERO Bridges are interconnected in a secured private BGP overlay
   routing instance to provide an OAL routing/bridging service that
   joins the underlying Internetworks of multiple disjoint
   administrative domains into a single unified OMNI link at a layer
   below IP. Each OMNI link instance is characterized by the set of
   Mobility Service Prefixes (MSPs) common to all mobile nodes. The

link extends to the point where a Relay is on the optimal route from
any correspondent node on the link, and provides a conduit between
the underlying Internetwork and the OMNI link. To the underlying
Internetwork, the Relay is the source of a route to the MSP, and
hence uplink traffic to the mobile node is naturally routed to the
nearest Relay.

AERO can be used with OMNI links that span private-use Internetworks
and/or public Internetworks such as the global Internet. In the
latter case, some end systems may be located behind global Internet
Network Address Translators (NATs). A means for robust traversal of
NATs while avoiding "triangle routing" is therefore provided.

AERO assumes the use of PIM Sparse Mode in support of multicast
communication. In support of Source Specific Multicast (SSM) when a
Mobile Node is the source, AERO route optimization ensures that a
shortest-path multicast tree is established with provisions for
mobility and multilink operation. In all other multicast scenarios
there are no AERO dependencies.

AERO was designed for aeronautical networking for both manned and
unmanned aircraft, where the aircraft is treated as a mobile node
that can connect an Internet of Things (IoT). AERO is also
applicable to a wide variety of other use cases. For example, it can
be used to coordinate the links of mobile nodes (e.g., cellphones,
tablets, laptop computers, etc.) that connect into a home enterprise
network via public access networks using tunneling software such as
OpenVPN [OVPN] with VPN or non-VPN services enabled according to the
appropriate security model. AERO can also be used to facilitate
terrestrial vehicular and urban air mobility (as well as pedestrian
communication services) for future intelligent transportation systems
[I-D.ietf-ipwave-vehicular-networking][I-D.templin-ipwave-uam-its].
Other applicable use cases are also in scope.

The following numbered sections present the AERO specification. The
appendices at the end of the document are non-normative.

2. Terminology

The terminology in the normative references applies; especially, the
terminology in the OMNI specification
[I-D.templin-6man-omni-interface] is used extensively throughout.
The following terms are defined within the scope of this document:

IPv6 Neighbor Discovery (ND)
   an IPv6 control message service for coordinating neighbor
   relationships between nodes connected to a common link. AERO uses
   the ND service specified in [RFC4861].

IPv6 Prefix Delegation
   a networking service for delegating IPv6 prefixes to nodes on the
   link. The nominal service is DHCPv6 [RFC8415], however alternate
   services (e.g., based on ND messaging) are also in scope. Most
   notably, a minimal form of prefix delegation known as "prefix
   registration" can be used if the Client knows its prefix in
   advance and can represent it in the IPv6 source address of an ND
   message.

Access Network (ANET)
   a node's first-hop data link service network (e.g., a radio access
   network, cellular service provider network, corporate enterprise
   network, etc.) that often provides link-layer security services
   such as IEEE 802.1X and physical-layer security (e.g., "protected
   spectrum") to prevent unauthorized access internally and with
   border network-layer security services such as firewalls and
   proxys that prevent unauthorized outside access.

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

Internetwork (INET)
   a connected IP network topology with a coherent routing and
   addressing plan and that provides a transit backbone service for
   ANET end systems. INETs also provide an underlay service over
   which the AERO virtual link is configured. Example INETs include
   corporate enterprise networks, aviation networks, and the public
   Internet itself. When there is no administrative boundary between
   an ANET and the INET, the ANET and INET are one and the same.

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

*NET
   a "wildcard" term referring to either ANET or INET when it is not
   necessary to draw a distinction between the two.

*NET interface
   a node's attachment to a link in a *NET.

*NET Partition
   frequently, *NETs such as large corporate enterprise networks are
   sub-divided internally into separate isolated partitions. Each
   partition is fully connected internally but disconnected from
   other partitions, and there is no requirement that separate
   partitions maintain consistent Internet Protocol and/or addressing
   plans. (Each *NET partition is seen as a separate OMNI link
   segment as discussed below.)

   *NET address
      an IP address assigned to a node's interface connection to a *NET.

   *NET encapsulation
      the encapsulation of a packet in an outer header or headers that
      can be routed within the scope of the local *NET partition.

   OMNI link
      the same as defined in [I-D.templin-6man-omni-interface], and
      manifested by IPv6 encapsulation [RFC2473]. The OMNI link spans
      underlying INET segments joined by virtual bridges in a spanning
      tree the same as a bridged campus LAN. AERO nodes on the OMNI
      link appear as single-hop neighbors even though they may be
      separated by multiple underlying INET hops, and can use Segment
      Routing [RFC8402] to cause packets to visit selected waypoints on
      the link.

   OMNI Interface
      a node's attachment to an OMNI link. Since the addresses assigned
      to an OMNI interface are managed for uniqueness, OMNI interfaces
      do not require Duplicate Address Detection (DAD) and therefore set
      the administrative variable 'DupAddrDetectTransmits' to zero
      [RFC4862].

   OMNI Adaptation Layer (OAL)
      an OMNI interface process whereby original IP packets admitted
      into the interface are wrapped in a mid-layer IPv6 header and
      subject to fragmentation and reassembly. The OAL is also
      responsible for generating MTU-related control messages as
      necessary, and for providing addressing context for spanning
      multiple segments of a bridged OMNI link.

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

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

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

   (OAL) 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.

   underlying interface
      a *NET interface over which an OMNI interface is configured.

   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:
      [I-D.templin-6man-omni-interface]). 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 delegated to an
      AERO Client or Relay.

   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
      [I-D.templin-6man-omni-interface].

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 [I-D.templin-6man-omni-interface].

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

AERO node
    a node that is connected to an OMNI link and participates in the
    AERO internetworking and mobility service.

AERO Client ("Client")
    an AERO node that connects over one or more underlying interfaces
    and requests MNP delegation/registration service from AERO Proxy/
    Servers. The Client assigns an MNP-LLA to the OMNI interface for
    use in ND exchanges with other AERO nodes and forwards original IP
    packets to correspondents according to OMNI interface neighbor
    cache state.

AERO Proxy/Server ("Proxy/Server")
    a dual-function node that provides a proxying service between AERO
    Clients and external peers on its Client-facing ANET interfaces
    (i.e., in the same fashion as for an enterprise network proxy) as
    well as default forwarding and Mobility Anchor Point (MAP)
    services for AERO Clients on its INET-facing interfaces. The
    Proxy/Server configures an OMNI interface and assigns an ADM-LLA
    to support the operation of IPv6 ND services, while advertising
    all of its associated MNPs via BGP peerings with Bridges. Note
    that the Proxy and Server functions can be considered logically
    separable, but since each Proxy/Server must be informed of all of
    the Client's other multilink Proxy/Server affiliations the AERO
    service is best supported when the two functions are coresident on
    the same physical or logical platform.

AERO Relay ("Relay")
    a Proxy/Server that provides forwarding services between nodes
    reached via the OMNI link and correspondents on connected
    downstream links. AERO Relays configure an OMNI interface and
    assign an ADM-LLA the same as Proxy/Servers. AERO Relays also run
    a dynamic routing protocol to discover any non-MNP IP GUA routes
    in service on its connected downstream network links. In both
    cases, the Relay advertises the MSP(s) to its downstream networks,
    and distributes all of its associated non-MNP IP GUA routes via
    BGP peerings with Bridges (i.e., the same as for Proxy/Servers).

AERO Bridge ("Bridge")
   a node that provides hybrid routing/bridging services (as well as
   a security trust anchor) for nodes on an OMNI link. The Bridge
   forwards carrier packets between OMNI link segments as OAL
   intermediate nodes without decrementing the network layer IP TTL/
   Hop Limit. AERO Bridges peer with Proxy/Servers and other Bridges
   to discover the full set of MNPs for the link as well as any non-
   MNP IP GUA routes that are reachable via Relays.

ingress tunnel endpoint (ITE)
   an OMNI interface endpoint that injects encapsulated packets into
   an OMNI link.

egress tunnel endpoint (ETE)
   an OMNI interface endpoint that receives encapsulated packets from
   an OMNI link.

link-layer address
   an IP address used as an encapsulation header source or
   destination address from the perspective of the OMNI interface.
   When an upper layer protocol (e.g., UDP) is used as part of the
   encapsulation, the port number is also considered as part of the
   link-layer address.

network layer address
   the source or destination address of an original IP packet
   presented to the OMNI interface.

end user network (EUN)
   an internal virtual or external edge IP network that an AERO
   Client or Relay connects to the rest of the network via the OMNI
   interface. The Client/Relay sees each EUN as a "downstream"
   network, and sees the OMNI interface as the point of attachment to
   the "upstream" network.

Mobile Node (MN)
   an AERO Client and all of its downstream-attached networks that
   move together as a single unit, i.e., an end system that connects
   an Internet of Things.

Mobile Router (MR)
   a MN's on-board router that forwards original IP packets between
   any downstream-attached networks and the OMNI link. The MR is the
   MN entity that hosts the AERO Client.

Route Optimization Source (ROS)

the AERO node nearest the source that initiates route
optimization. The ROS may be a Proxy/Server acting on behalf of
the source Client, or may be the Client itself.

Route Optimization responder (ROR)
   the AERO node nearest the target destination that responds to
   route optimization requests. The ROR may be a Proxy/Server acting
   on behalf of a target MNP Client, a Relay for a non-MNP
   destination or may be the target Client itself.

MAP List
   a geographically and/or topologically referenced list of addresses
   of all Proxy/Servers within the same OMNI link. There is a single
   MAP list for the entire OMNI link.

Distributed Mobility Management (DMM)
   a BGP-based overlay routing service coordinated by Proxy/Servers
   and Bridges that tracks all Proxy/Server-to-Client associations.

Mobility Service (MS)
   the collective set of all Proxy/Servers, Bridges and Relays that
   provide the AERO Service to Clients.

Mobility Service Endpoint MSE)
   an individual Proxy/Server, Bridge or Relay in the Mobility
   Service.

Throughout the document, the simple terms "Client", "Proxy/Server",
"Bridge" and "Relay" refer to "AERO Client", "AERO Proxy/Server",
"AERO Bridge" and "AERO Relay", respectively. Capitalization is used
to distinguish these terms from other common Internetworking uses in
which they appear without capitalization.

The terminology of DHCPv6 [RFC8415] and IPv6 ND [RFC4861] (including
the names of node variables, messages and protocol constants) is used
throughout this document. The terms "All-Routers multicast", "All-
Nodes multicast", "Solicited-Node multicast" and "Subnet-Router
anycast" are defined in [RFC4291]. Also, the term "IP" is used to
generically refer to either Internet Protocol version, i.e., IPv4
[RFC0791] or IPv6 [RFC8200].

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

3. **Automatic Extended Route Optimization (AERO)**

   The following sections specify the operation of IP over OMNI links
   using the AERO service:

3.1. **AERO Node Types**

   AERO Clients are Mobile Nodes (MNs) that configure OMNI interfaces
   over underlying interfaces with addresses that may change when the
   Client moves to a new network connection point. AERO Clients
   register their Mobile Network Prefixes (MNPs) with the AERO service,
   and distribute the MNPs to nodes on EUNs. AERO Bridges, Proxy/
   Servers and Relays are critical infrastructure elements in fixed
   (i.e., non-mobile) INET deployments and hence have permanent and
   unchanging INET addresses. Together, they constitute the AERO
   service which provides an OMNI link virtual overlay for connecting
   AERO Clients.

   AERO Bridges provide hybrid routing/bridging services (as well as a
   security trust anchor) for nodes on an OMNI link. Bridges use
   standard IPv6 routing to forward carrier packets both within the same
   *NET partition and between disjoint *NET partitions based on an IPv6
   encapsulation mid-layer known as the OMNI Adaptation Layer (OAL)
   [I-D.templin-6man-omni-interface]. During forwarding, the inner IP
   layer experiences a virtual bridging service since the inner IP TTL/
   Hop Limit is not decremented. Each Bridge also peers with Proxy/
   Servers and other Bridges in a dynamic routing protocol instance to
   provide a Distributed Mobility Management (DMM) service for the list
   of active MNPs (see Section 3.2.3). Bridges present the OMNI link as
   a set of one or more Mobility Service Prefixes (MSPs) and configure
   secured tunnels with Proxy/Servers, Relays and other Bridges; they
   further maintain IP forwarding table entries for each MNP and any
   other reachable non-MNP prefixes.

   AERO Proxy/Servers in distributed INET locations provide default
   forwarding and mobility/multilink services for AERO Client Mobile
   Nodes (MNs). Each Proxy/Server also peers with Bridges in a dynamic
   routing protocol instance to advertise its list of associated MNPs
   (see Section 3.2.3). Proxy/Servers facilitate prefix delegation/
   registration exchanges with Clients, where each delegated prefix
   becomes an MNP taken from an MSP. Proxy/Servers forward carrier
   packets between OMNI interface neighbors and track each Client's
   mobility profiles. Proxy/Servers provide a conduit for ANET Clients
   to associate with additional Proxy/Servers in external INETs. The
   Proxy forwards original IP packets between Clients and the OMNI link
   according to forwarding information in the neighbor cache.

AERO Relays are Proxy/Servers that provide forwarding services to
exchange original IP packets between the OMNI interface and INET/EUN
interfaces. Relays are provisioned with MNPs the same as for an AERO
Client, and also run a dynamic routing protocol to discover any non-
MNP IP routes. The Relay advertises the MSP(s) to its connected
networks, and distributes all of its associated MNPs and non-MNP
routes via BGP peerings with Bridges

3.2. The AERO Service over OMNI Links

3.2.1. AERO/OMNI Reference Model

Figure 1 presents the basic OMNI link reference model:

```
                        +----------------+
                        | AERO Bridge B1 |
                        | Nbr: S1, S2, P1|
                        |(X1->S1; X2->S2)|
                        |      MSP M1     |
                        +-+------------+-+
    +--------------+      | Secured  |      +--------------+
    | AERO P/S S1 |      | tunnels  |      | AERO P/S S2 |
    | Nbr: C1, B1 +-----+          +-----+ Nbr: C2, B1 |
    | default->B1 |                       | default->B1 |
    |    X1->C1    |                       |    X2->C2    |
    +-------+------+                       +------+-------+
            |       OMNI link                     |
     X===+===+=====================================+===+===X
         |                                             |
    +-----+--------+                         +--------+-----+
    |AERO Client C1|                         |AERO Client C2|
    |    Nbr: S1   |                         |    Nbr: S2    |
    | default->S1  |                         | default->S2  |
    |    MNP X1    |                         |    MNP X2     |
    +------+-------+                         +-----+--------+
           |                                        |
         .-.                                      .-.
       ,-(  _)-.                               ,-( _)-.
     .-(_  IP    )-.    +-------+     +-------+  .-(_  IP    )-.
    (__    EUN        )--|Host H1|    |Host H2|--(__    EUN        )
      `-(_____)-'      +-------+     +-------+    `-(_____)-'
```
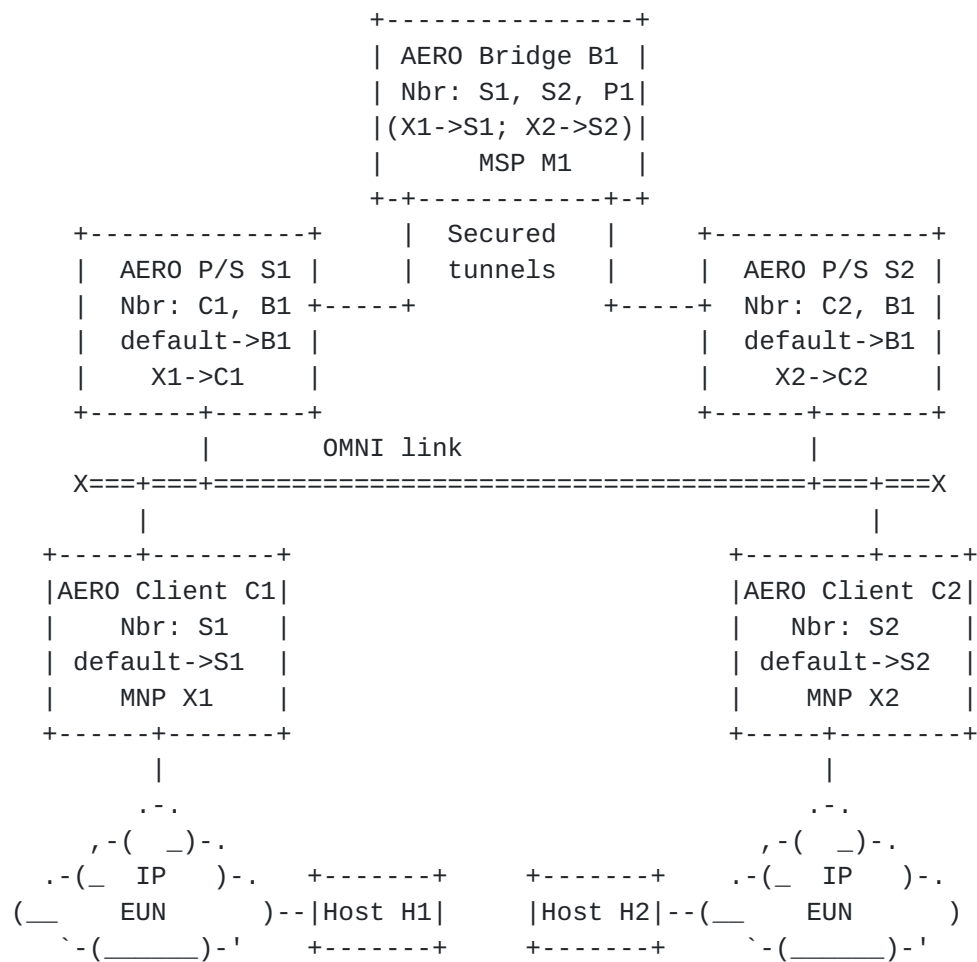
                   Figure 1: AERO/OMNI Reference Model

In this model:

o   the OMNI link is an overlay network service configured over one or
    more underlying *NET partitions which may be managed by different

administrative authorities and have incompatible protocols and/or addressing plans.

o AERO Bridge B1 aggregates Mobility Service Prefix (MSP) M1, discovers Mobile Network Prefixes (MNPs) X* and advertises the MSP via BGP peerings over secured tunnels to Proxy/Servers (S1, S2). Bridges connect the disjoint segments of a partitioned OMNI link.

o AERO Proxy/Servers S1 and S2 configure secured tunnels with Bridge B1 and also provide mobility, multilink and default router services for their associated Clients C1 and C2.

o AERO Clients C1 and C2 associate with Proxy/Servers S1 and S2, respectively. They receive Mobile Network Prefix (MNP) delegations X1 and X2, and also act as default routers for their associated physical or internal virtual EUNs. Simple hosts H1 and H2 attach to the EUNs served by Clients C1 and C2, respectively.

An OMNI link configured over a single *NET appears as a single unified link with a consistent underlying network addressing plan. In that case, all nodes on the link can exchange carrier packets via simple *NET encapsulation, since the underlying *NET is connected. In common practice, however, an OMNI link may be partitioned into multiple "segments", where each segment is a distinct *NET potentially managed under a different administrative authority (e.g., as for worldwide aviation service providers such as ARINC, SITA, Inmarsat, etc.). Individual *NETs may also themselves be partitioned internally, in which case each internal partition is seen as a separate segment.

The addressing plan of each segment is consistent internally but will often bear no relation to the addressing plans of other segments. Each segment is also likely to be separated from others by network security devices (e.g., firewalls, proxys, packet filtering gateways, etc.), and in many cases disjoint segments may not even have any common physical link connections. Therefore, nodes can only be assured of exchanging carrier packets directly with correspondents in the same segment, and not with those in other segments. The only means for joining the segments therefore is through inter-domain peerings between AERO Bridges.

The same as for traditional campus LANs, multiple OMNI link segments can be joined into a single unified link via a virtual bridging service using the OMNI Adaptation Layer (OAL) [I-D.templin-6man-omni-interface] which inserts a mid-layer IPv6 encapsulation header that supports inter-segment forwarding (i.e., bridging) without decrementing the network-layer TTL/Hop Limit. This bridging of OMNI link segments is shown in Figure 2:

```
           . . . . . . . . . . . . . . . . . . . . . . . . . .
         .                                                       .
         .                   .-(:::::::::)                       .
         .               .-(:::::::::::::)-.    +-+              .
         .              (:::: Segment A :::)--|B|---+           .
         .               `-(:::::::::::::)-'   +-+   |          .
         .                  `-(:::::::)-'             |          .
         .                                            |          .
         .                   .-(:::::::::)            |          .
         .               .-(:::::::::::::)-.    +-+   |          .
         .              (:::: Segment B :::)--|B|---+           .
         .               `-(:::::::::::::)-'   +-+   |          .
         .                  `-(:::::::)-'             |          .
         .                                            |          .
         .                   .-(:::::::::)            |          .
         .               .-(:::::::::::::)-.    +-+   |          .
         .              (:::: Segment C :::)--|B|---+           .
         .               `-(:::::::::::::)-'   +-+   |          .
         .                  `-(:::::::)-'             |          .
         .                                            |          .
         .                     ..(etc)..              X          .
         .                                                       .
         .                                                       .
         .     <- OMNI link Bridged by encapsulation ->   .
           . . . . . . . . . . . . . . . .. . . . . . . . .
```
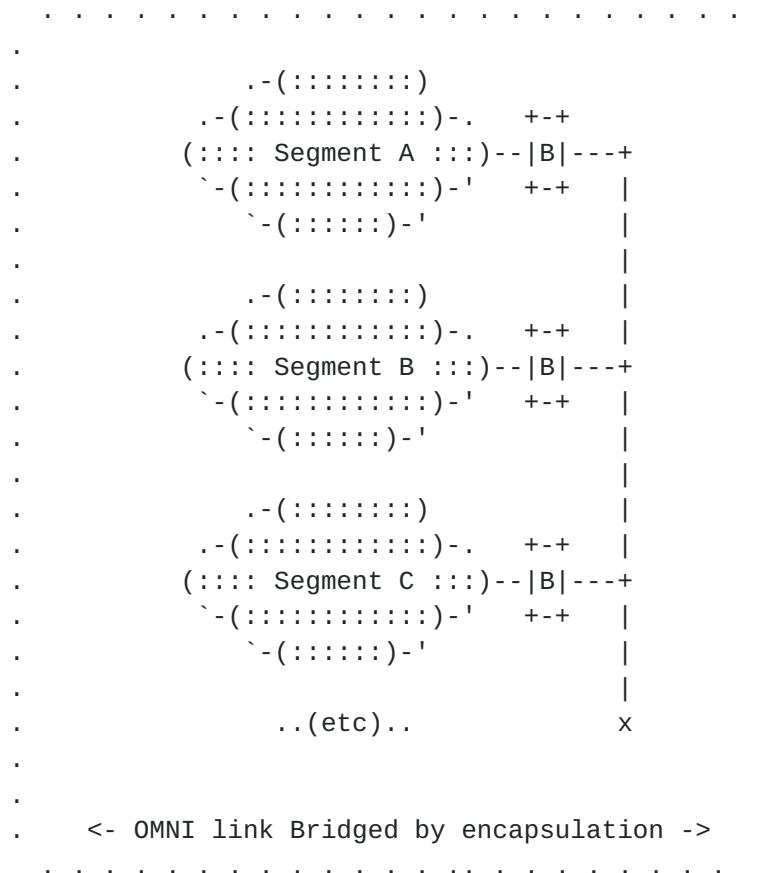
                   Figure 2: Bridging OMNI Link Segments

   Bridges, Proxy/Servers and Relays connect via secured INET tunnels
   over their respective segments in a spanning tree topology rooted at
   the Bridges. The secured spanning tree supports strong
   authentication for IPv6 ND control messages and may also be used to
   convey the initial carrier packets in a flow. Route optimization can
   then be employed to cause carrier packets to take more direct paths
   between OMNI link neighbors without having to strictly follow the
   spanning tree.

3.2.2. Addressing and Node Identification

   AERO nodes on OMNI links use the Link-Local Address (LLA) prefix
   fe80::/64 [RFC4291] to assign LLAs used for network-layer addresses
   in link-scoped IPv6 ND and data messages. AERO Clients use LLAs
   constructed from MNPs (i.e., "MNP-LLAs") while other AERO nodes use
   LLAs constructed from administrative identification values ("ADM-
   LLAs") as specified in [I-D.templin-6man-omni-interface].

   AERO nodes also use the Unique Local Address (ULA) prefix fd00::/8
   followed by a pseudo-random 40-bit OMNI domain identifier to form the

prefix [ULA]::/48, then include a 16-bit OMNI link identifier '*' to
form the prefix [ULA*]::/64 [RFC4291]. The AERO node then uses the
prefix [ULA*]::/64 to form "MNP-ULAs" or "ADM-ULA"s as specified in
[I-D.templin-6man-omni-interface] to support OAL addressing. AERO
Clients also use Temporary ULAs constructed per
[I-D.templin-6man-omni-interface], where the addresses are typically
used only in initial control message exchanges until a stable MNP-
LLA/ULA is assigned.

AERO MSPs, MNPs and non-MNP routes are typically based on Global
Unicast Addresses (GUAs), but in some cases may be based on private-
use addresses. See [I-D.templin-6man-omni-interface] for a full
specification of LLAs, ULAs and GUAs used by AERO nodes on OMNI
links.

Finally, AERO Clients and Proxy/Servers configure node identification
values as specified in [I-D.templin-6man-omni-interface].

3.2.3. AERO Routing System

The AERO routing system comprises a private instance of the Border
Gateway Protocol (BGP) [RFC4271] that is coordinated between Bridges
and Proxy/Servers and does not interact with either the public
Internet BGP routing system or any underlying INET routing systems.

In a reference deployment, each Proxy/Server is configured as an
Autonomous System Border Router (ASBR) for a stub Autonomous System
(AS) using a 32-bit AS Number (ASN) [RFC4271] that is unique within
the BGP instance, and each Proxy/Server further uses eBGP to peer
with one or more Bridges but does not peer with other Proxy/Servers.
Each *NET of a multi-segment OMNI link must include one or more
Bridges, which peer with the Proxy/Servers within that *NET. All
Bridges within the same *NET are members of the same hub AS, and use
iBGP to maintain a consistent view of all active routes currently in
service. The Bridges of different *NETs peer with one another using
eBGP.

Bridges maintain forwarding table entries only for the ULAs
corresponding to MNP and non-MNP routes that are currently active,
and carrier packets destined to all other ULAs will correctly incur
Destination Unreachable messages due to the black-hole route. In
this way, Proxy/Servers and Relays have only partial topology
knowledge (i.e., they know only about the routes their directly
associated Clients and non-AERO links) and they forward all other
carrier packets to Bridges which have full topology knowledge.

Each OMNI link segment assigns a unique ADM-ULA sub-prefix of
[ULA*]::/96. For example, a first segment could assign

[ULA*]::1000/116, a second could assign [ULA*]::2000/116, a third
could assign [ULA*]::3000/116, etc. Within each segment, each Proxy/
Server configures an ADM-ULA within the segment's prefix, e.g., the
Proxy/Servers within [ULA*]::2000/116 could assign the ADM-ULAs
[ULA*]::2011/116, [ULA*]::2026/116, [ULA*]::2003/116, etc.

The administrative authorities for each segment must therefore
coordinate to assure mutually-exclusive ADM-ULA prefix assignments,
but internal provisioning of ADM-ULAs an independent local
consideration for each administrative authority. For each ADM-ULA
prefix, the Bridge(s) that connect that segment assign the all-zero's
address of the prefix as a Subnet Router Anycast address. For
example, the Subnet Router Anycast address for [ULA*]::1023/116 is
simply [ULA*]::1000.

ADM-ULA prefixes are statically represented in Bridge forwarding
tables. Bridges join multiple segments into a unified OMNI link over
multiple diverse administrative domains. They support a bridging
function by first establishing forwarding table entries for their
ADM-ULA prefixes either via standard BGP routing or static routes.
For example, if three Bridges ('A', 'B' and 'C') from different
segments serviced [ULA*]::1000/116, [ULA*]::2000/116 and
[ULA*]::3000/116 respectively, then the forwarding tables in each
Bridge are as follows:

A: [ULA*]::1000/116->local, [ULA*]::2000/116->B, [ULA*]::3000/116->C

B: [ULA*]::1000/116->A, [ULA*]::2000/116->local, [ULA*]::3000/116->C

C: [ULA*]::1000/116->A, [ULA*]::2000/116->B, [ULA*]::3000/116->local

These forwarding table entries are permanent and never change, since
they correspond to fixed infrastructure elements in their respective
segments.

MNP ULAs are instead dynamically advertised in the AERO routing
system by Proxy/Servers and Relays that provide service for their
corresponding MNPs. For example, if three Proxy/Servers ('D', 'E'
and 'F') service the MNPs 2001:db8:1000:2000::/56,
2001:db8:3000:4000::/56 and 2001:db8:5000:6000::/56 then the routing
system would include:

D: [ULA*]:2001:db8:1000:2000/120

E: [ULA*]:2001:db8:3000:4000/120

F: [ULA*]:2001:db8:5000:6000/120

A full discussion of the BGP-based routing system used by AERO is
found in [I-D.ietf-rtgwg-atn-bgp].

3.2.4. OMNI Link Segment Routing

With the Client and partition prefixes in place in Bridge forwarding
tables, the OMNI interface sends control and data messages toward
AERO destination nodes located in different OMNI link segments over
the spanning tree. The OMNI interface uses the OMNI Adaptation Layer
(OAL) encapsulation service [I-D.templin-6man-omni-interface], and
includes an OMNI Routing Header (ORH) as an extension to the OAL
header if final segment forwarding information is available, e.g., in
the neighbor cache. (For nodes located in the same OMNI link
segment, or when no final segment forwarding information is
available, the ORH may be omitted.) The ORH is formatted as shown in
Figure 3:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Next Header  |  Hdr Ext Len  |  Routing Type |   SRT   | FMT |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Last Hop Segment-id (LHS)                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                   Link Layer Address (L2ADDR)                ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                 Destination Suffix (if necessary)            ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                   Null Padding (if necessary)                ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
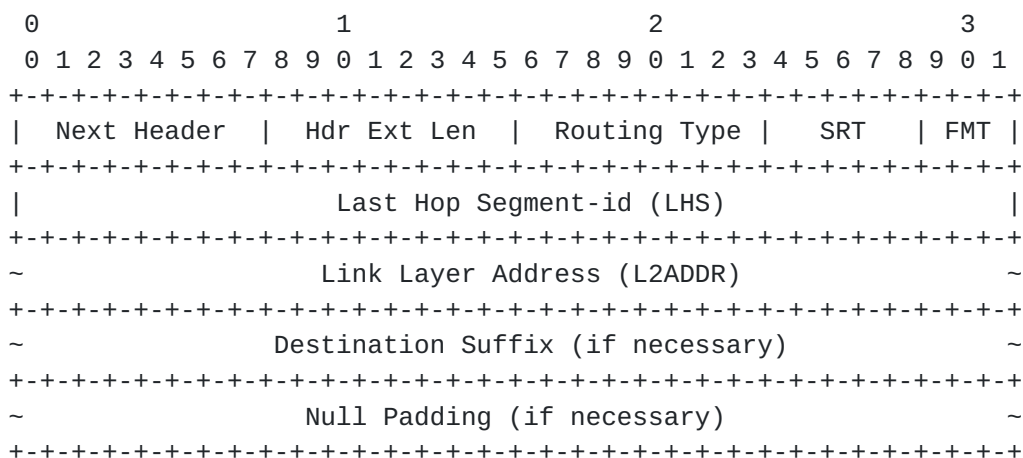
                 Figure 3: OMNI Routing Header (ORH) Format

In this format:

o Next Header identifies the type of header immediately following
   the ORH.

o Hdr Ext Len is the length of the Routing header in 8-octet units
   (not including the first 8 octets), with trailing padding added if
   necessary to produce an integral number of 8-octet units.

o Routing Type is set to TBD1 (see IANA Considerations).

o Segments Left is omitted, and replaced by a 5-bit SRT and 3-bit
   FMT field.

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

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

   * When the most significant bit (i.e., "Framework") is set to 1,
      L2ADDR is the *NET encapsulation address for the target Client
      itself; otherwise L2ADDR is the address of the Proxy/Server
      named in the LHS.

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

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

o LHS - the 32 bit ID of a node in the Last Hop Segment that
   services the target. When SRT and LHS are both set to 0, the LHS
   is considered unspecified. When SRT is set to 0 and LHS is non-
   zero, the prefix length is set to 128. SRT and LHS provide
   guidance to the OMNI interface forwarding algorithm.
   Specifically, if SRT/LHS is located in the local OMNI link
   segment, the OAL source can omit the ORH and (following any
   necessary NAT traversal messaging) send directly to the OAL
   destination according to FMT/L2ADDR. Otherwise, it includes the
   ORH and forwards according to the OMNI link spanning tree.

o Link Layer Address (L2ADDR) - Formatted according to FMT, and
   identifies the link-layer address (i.e., the encapsulation
   address) of the target. The UDP Port Number appears in the first
   two 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 *NET encapsulation
   address for local forwarding when SRT/LHS is located in the same
   OMNI link segment. Note that if the target is behind a NAT,
   L2ADDR will contain the mapped *NET address stored in the NAT;
   otherwise, L2ADDR will contain the native *NET information of the
   target itself.

o Destination Suffix is a 64-bit field included only for OAL non-
   first-fragments. Present only when Hdr Ext Len indicates that at
   least 8 bytes follow L2ADDR. When present, encodes the 64-bit
   MNP-ULA suffix for the target Client.

o Null Padding contains zero-valued octets as necessary to pad the
   ORH to an integral number of 8-octet units.

AERO neighbors use OAL encapsulation and fragmentation to exchange
OAL packets as specified in [I-D.templin-6man-omni-interface]. When
an AERO node's OMNI interface (acting as an OAL source) uses OAL
encapsulation for an original IP packet with source address
2001:db8:1:2::1 and destination address 2001:db8:1234:5678::1, it
sets the OAL header source address to its own ULA (e.g.,
[ULA*]::2001:db8:1:2), sets the destination address to the MNP-ULA
corresponding to the IP destination address (e.g.,
[ULA*]::2001:db8:1234:5678), selects an Identification and appends an
OAL checksum.

If the neighbor cache information indicates that the target is in a
different segment, the OAL source next inserts an ORH immediately
following the OAL header while including the correct SRT, FMT, LHS,
L2ADDR and Destination Suffix if fragmentation if needed (in this
case, the Destination Suffix is 2001:db8:1234:5678). Next, the OAL
source overwrites the OAL header destination address with the LHS
Subnet Router Anycast address (for example, for LHS 3000:4567 with
SRT 16, the Subnet Router Anycast address is [ULA*]::3000:0000).
(Note: if the ADM-ULA of the last-hop Proxy/Server is known but the
SRT, FMT, LHS and L2ADDR are not (yet) known, the OAL source instead
overwrites the OAL header destination address with the ADM-ULA.)

The OAL source then fragments the OAL packet, with each resulting OAL
fragment including the OAL/ORH headers while only the first fragment
includes the original IPv6 header. (Note that the packet is prepared
as an "atomic" OAL fragment even if no actual fragmentation was
required.) The OAL source finally encapsulates each resulting OAL
fragment in an *NET header to form an OAL carrier packet, with source
address set to its own *NET address (e.g., 192.0.2.100) and
destination set to the *NET address of a Bridge (e.g., 192.0.2.1).

The carrier packet encapsulation format in the above example is shown
in Figure 4:

```
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |            *NET Header         |
      |          src = 192.0.2.100     |
      |           dst = 192.0.2.1      |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |           OAL IPv6 Header      |
      |   src = [ULA*]::2001:db8:1:2   |
      |     dst= [ULA*]::3000:0000     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |         ORH (if necessary)     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |        OAL Fragment Header     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |         Original IP Header     |
      |        (first-fragment only)   |
      |       src = 2001:db8:1:2::1    |
      |    dst = 2001:db8:1234:5678::1 |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                               |
      ~                               ~
      ~ Original Packet Body/Fragment ~
      ~                               ~
      |                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                    Figure 4: Carrier Packet Format

   In this format, the original IP header and packet body/fragment are
   from the original IP packet, the OAL header is an IPv6 header
   prepared according to [RFC2473], the ORH is a Routing Header
   extension of the OAL header the Fragment Header identifies each
   fragment, and the INET header is prepared as discussed in
   Section 3.6. When the OAL source transmits the resulting carrier
   packets, they are forwarded over possibly multiple OMNI link spanning
   tree hops until they arrive at the OAL destination.

   This gives rise to a routing system that contains both Client MNP-ULA
   routes that may change dynamically due to regional node mobility and
   per-partition ADM-ULA routes that rarely if ever change. The Bridges
   can therefore provide link-layer bridging by sending carrier packets
   over the spanning tree instead of network-layer routing according to
   MNP routes. As a result, opportunities for loss due to node mobility
   between different segments are mitigated.

   In normal operations, IPv6 ND messages are conveyed over secured
   paths between OMNI link neighbors so that specific Proxy/Servers or
   Relays can be addressed without being subject to mobility events.
   Conversely, only the first few carrier packets destined to Clients

need to traverse secured paths until route optimization can determine
a more direct path.

Note: When the OAL source and destination are on the same INET
segment, the ORH is not needed and OAL header compression can be used
to significantly reduce encapsulation overhead
[I-D.templin-6man-omni-interface].

Note: When the OAL source has multiple original IP packets to send to
the same OAL destination, it can perform "packing" to generate a
"super-packet" [I-D.templin-6man-omni-interface]. In that case, the
OAL/ORH super-packet may include up to N original IP packets as long
as the total length of the super-packet does not exceed the OMNI
interface MTU.

Note: Use of an IPv6 "minimal encapsulation" format (i.e., an IPv6
variant of [RFC2004]) based on extensions to the ORH was considered
and abandoned. In the approach, the ORH would be inserted as an
extension header to the original IPv6 packet header. The IPv6
destination address would then be written into the ORH, and the ULA
corresponding to the destination would be overwritten in the IPv6
destination address. This would seemingly convey enough forwarding
information so that OAL encapsulation could be avoided. However,
this "minimal encapsulation" IPv6 packet would then have a non-ULA
source address and ULA destination address, an incorrect value in
upper layer protocol checksums, and a Hop Limit that is decremented
within the spanning tree when it should not be. The insertion and
removal of the ORH would also entail rewriting the Payload Length and
Next Header fields - again, invalidating upper layer checksums.
These irregularities would result in implementation challenges and
the potential for operational issues, e.g., since actionable ICMPv6
error reports could not be delivered to the original source. In
order to address the issues, still more information such as the
original IPv6 source address could be written into the ORH. However,
with the additional information the benefit of the "minimal
encapsulation" savings quickly diminishes, and becomes overshadowed
by the implementation and operational irregularities.

3.2.5. Segment Routing Topologies (SRTs)

The 64-bit sub-prefixes of [ULA]::/48 identify up to 2^16 distinct
Segment Routing Topologies (SRTs). Each SRT is a mutually-exclusive
OMNI link overlay instance using a distinct set of ULAs, and emulates
a Virtual LAN (VLAN) service for the OMNI link. In some cases (e.g.,
when redundant topologies are needed for fault tolerance and
reliability) it may be beneficial to deploy multiple SRTs that act as
independent overlay instances. A communication failure in one
instance therefore will not affect communications in other instances.

Each SRT is identified by a distinct value in bits 48-63 of
[ULA]::/48, i.e., as [ULA0]::/64, [ULA1]::/64, [ULA2]::/64, etc.
Each OMNI interface is identified by a unique interface name (e.g.,
omni0, omni1, omni2, etc.) and assigns an anycast ADM-ULA
corresponding to its SRT prefix length. The anycast ADM-ULA is used
for OMNI interface determination in Safety-Based Multilink (SBM) as
discussed in [I-D.templin-6man-omni-interface]. Each OMNI interface
further applies Performance-Based Multilink (PBM) internally.

3.2.6. Segment Routing For OMNI Link Selection

An original IPv6 source can direct an IPv6 packet to an AERO node by
including a standard IPv6 Segment Routing Header (SRH) [RFC8754] with
the anycast ADM-ULA for the selected SRT as either the IPv6
destination or as an intermediate hop within the SRH. This allows
the original source to determine the specific OMNI link topology an
original IPv6 packet will traverse when there may be multiple
alternatives.

When the AERO node processes the SRH and forwards the original IPv6
packet to the correct OMNI interface, the OMNI interface writes the
next IPv6 address from the SRH into the IPv6 destination address and
decrements Segments Left. If decrementing would cause Segments Left
to become 0, the OMNI interface deletes the SRH before forwarding.
This form of Segment Routing supports Safety-Based Multilink (SBM).

3.2.7. Segment Routing Within the OMNI Link

OAL sources can insert an ORH for Segment Routing within the OMNI
link to influence the paths of OAL packets sent to OAL destinations
in remote segments without requiring all carrier packets to traverse
strict spanning tree paths.

When an AERO node's OMNI interface has an original IP packet to send
to a target discovered through route optimization located in the same
OMNI link segment, it acts as an OAL source to perform OAL
encapsulation and fragmentation. The node then uses the target's
Link Layer Address (L2ADDR) information for *NET encapsulation.

When an AERO node's OMNI interface has an original IP packet to send
to a route optimization target located in a remote OMNI link segment,
it acts as an OAL source the same as above but also includes an ORH
while setting the OAL destination to the Subnet Router Anycast
address for the final OMNI link segment, then forwards the resulting
carrier packets to a Bridge.

When a Bridge receives a carrier packet destined to its Subnet Router
Anycast address with an ORH with SRT/LHS values corresponding to the

local segment, it examines the L2ADDR according to FMT and removes
the ORH from the carrier packet. The Bridge then writes the MNP-ULA
corresponding to the ORH Destination Suffix into the OAL destination
address, re-encapsulates the carrier packet according to L2ADDR and
forwards the carrier packet either to the LHS Proxy/Server or
directly to the target Client itself. In this way, the Bridge
participates in route optimization to reduce traffic load and
suboptimal routing through strict spanning tree paths.

3.3. OMNI Interface Characteristics

OMNI interfaces are virtual interfaces configured over one or more
underlying interfaces classified as follows:

o INET interfaces connect to an INET either natively or through one
   or several NATs. Native INET interfaces have global IP addresses
   that are reachable from any INET correspondent. The INET-facing
   interfaces of Proxy/Servers are native interfaces, as are Relay
   and Bridge interfaces. NATed INET interfaces connect to a private
   network behind one or more NATs that provide INET access. Clients
   that are behind a NAT are required to send periodic keepalive
   messages to keep NAT state alive when there are no carrier packets
   flowing.

o ANET interfaces connect to an ANET that is separated from the open
   INET by a Proxy/Server. Proxy/Servers can actively issue control
   messages over the INET on behalf of the Client to reduce ANET
   congestion.

o VPNed interfaces use security encapsulation over the INET to a
   Virtual Private Network (VPN) server that also acts as a Proxy/
   Server. Other than the link-layer encapsulation format, VPNed
   interfaces behave the same as Direct interfaces.

o Direct (i.e., single-hop point-to-point) interfaces connect a
   Client directly to a Proxy/Server without crossing any ANET/INET
   paths. An example is a line-of-sight link between a remote pilot
   and an unmanned aircraft. The same Client considerations apply as
   for VPNed interfaces.

OMNI interfaces use OAL encapsulation and fragmentation as discussed
in Section 3.2.4. OMNI interfaces use *NET encapsulation (see:
Section 3.6) to exchange carrier packets with OMNI link neighbors
over INET or VPNed interfaces as well as over ANET interfaces for
which the Client and Proxy/Server may be multiple IP hops away. OMNI
interfaces do not use link-layer encapsulation over Direct underlying
interfaces or ANET interfaces when the Client and Proxy/Server are
known to be on the same underlying link.

OMNI interfaces maintain a neighbor cache for tracking per-neighbor
state the same as for any interface. OMNI interfaces use ND messages
including Router Solicitation (RS), Router Advertisement (RA),
Neighbor Solicitation (NS) and Neighbor Advertisement (NA) for
neighbor cache management.

OMNI interfaces send ND messages with an OMNI option formatted as
specified in [I-D.templin-6man-omni-interface]. The OMNI option
includes prefix registration information and Interface Attributes
containing link information parameters for the OMNI interface's
underlying interfaces. Each OMNI option may include multiple
Interface Attributes sub-options, each identified by an ifIndex
value.

A Client's OMNI interface may be configured over multiple underlying
interface connections. For example, common mobile handheld devices
have both wireless local area network ("WLAN") and cellular wireless
links. These links are often used "one at a time" with low-cost WLAN
preferred and highly-available cellular wireless as a standby, but a
simultaneous-use capability could provide benefits. In a more
complex example, aircraft frequently have many wireless data link
types (e.g. satellite-based, cellular, terrestrial, air-to-air
directional, etc.) with diverse performance and cost properties.

If a Client's multiple underlying interfaces are used "one at a time"
(i.e., all other interfaces are in standby mode while one interface
is active), then ND message OMNI options include only a single
Interface Attributes sub-option set to constant values. In that
case, the Client would appear to have a single interface but with a
dynamically changing link-layer address.

If the Client has multiple active underlying interfaces, then from
the perspective of ND it would appear to have multiple link-layer
addresses. In that case, ND message OMNI options MAY include
multiple Interface Attributes sub-options - each with values that
correspond to a specific interface. Every ND message need not
include Interface Attributes for all underlying interfaces; for any
attributes not included, the neighbor considers the status as
unchanged.

Bridge and Proxy/Server OMNI interfaces may be configured over one or
more secured tunnel interfaces. The OMNI interface configures both
an ADM-LLA and its corresponding ADM-ULA, while the underlying
secured tunnel interfaces are either unnumbered or configure the same
ULA. The OMNI interface acting as an OAL source encapsulates and
fragments each original IP packet, then and presents the resulting
carrier packets to the underlying secured tunnel interface. Routing
protocols such as BGP that run over the OMNI interface do not employ

OAL encapsulation, but rather present the routing protocol messages
directly to the underlying secured tunnels while using the ULA as the
source address. This distinction must be honored consistently
according to each node's configuration so that the IP forwarding
table will associate discovered IP routes with the correct interface.

3.4. OMNI Interface Initialization

AERO Proxy/Servers and Clients configure OMNI interfaces as their
point of attachment to the OMNI link. AERO nodes assign the MSPs for
the link to their OMNI interfaces (i.e., as a "route-to-interface")
to ensure that original IP packets with destination addresses covered
by an MNP not explicitly assigned to a non-OMNI interface are
directed to the OMNI interface.

OMNI interface initialization procedures for Proxy/Servers, Clients
and Bridges are discussed in the following sections.

3.4.1. AERO Proxy/Server and Relay Behavior

When a Proxy/Server enables an OMNI interface, it assigns an
ADM-{LLA,ULA} appropriate for the given OMNI link segment. The
Proxy/Server also configures secured tunnels with one or more
neighboring Bridges and engages in a BGP routing protocol session
with each Bridge.

The OMNI interface provides a single interface abstraction to the IP
layer, but internally comprises multiple secured tunnels as well as
an NBMA nexus for sending carrier packets to OMNI interface
neighbors. The Proxy/Server further configures a service to
facilitate ND exchanges with AERO Clients and manages per-Client
neighbor cache entries and IP forwarding table entries based on
control message exchanges.

Relays are simply Proxy/Servers that run a dynamic routing protocol
to redistribute routes between the OMNI interface and INET/EUN
interfaces (see: Section 3.2.3). The Relay provisions MNPs to
networks on the INET/EUN interfaces (i.e., the same as a Client would
do) and advertises the MSP(s) for the OMNI link over the INET/EUN
interfaces. The Relay further provides an attachment point of the
OMNI link to a non-MNP-based global topology.

3.4.2. AERO Client Behavior

When a Client enables an OMNI interface, it assigns either an
MNP-{LLA, ULA} or a Temporary ULA and sends RS messages with ND
parameters over its underlying interfaces to a Proxy/Server, which
returns an RA message with corresponding parameters. The RS/RA

messages may pass through one or more NATs in the case of a Client's
INET interface. (Note: if the Client used a Temporary ULA in its
initial RS message, it will discover an MNP-{LLA, ULA} in the
corresponding RA that it receives from the Proxy/Server and begin
using these new addresses. If the Client is operating outside the
context of AERO infrastructure such as in a Mobile Ad-hoc Network
(MANET), however, it may continue using Temporary ULAs for Client-to-
Client communications until it encounters an infrastructure element
that can provide an MNP.)

3.4.3. AERO Bridge Behavior

AERO Bridges configure an OMNI interface and assign the ADM-ULA
Subnet Router Anycast address for each OMNI link segment they connect
to. Bridges configure secured tunnels with Proxy/Servers and other
Bridges, and engage in a BGP routing protocol session with neighbors
on the spanning tree (see: Section 3.2.3).

3.5. OMNI Interface Neighbor Cache Maintenance

Each OMNI interface maintains a conceptual neighbor cache that
includes an entry for each neighbor it communicates with on the OMNI
link per [RFC4861]. In addition to ordinary neighbor cache entries,
Proxy neighbor cache entries are created and maintained by AERO
Proxy/Servers when they proxy Client ND message exchanges [RFC4389].
AERO Proxy/Servers maintain proxy neighbor cache entries for each of
their associated Clients.

To the list of neighbor cache entry states in Section 7.3.2 of
[RFC4861], Proxy/Server OMNI interfaces add an additional state
DEPARTED that applies to Clients that have recently departed. The
interface sets a "DepartTime" variable for the neighbor cache entry
to "DEPART_TIME" seconds. DepartTime is decremented unless a new ND
message causes the state to return to REACHABLE. While a neighbor
cache entry is in the DEPARTED state, carrier packets destined to the
target Client are forwarded to the Client's new location instead of
being dropped. When DepartTime decrements to 0, the neighbor cache
entry is deleted. It is RECOMMENDED that DEPART_TIME be set to the
default constant value REACHABLE_TIME plus 10 seconds (40 seconds by
default) to allow a window for carrier packets in flight to be
delivered while stale route optimization state may be present.

Proxy/Servers can act as RORs on behalf of disadvantaged Clients
according to the Proxy Neighbor Advertisement specification in
Section 7.2.8 of [RFC4861]. (Well-connected Clients can act as an
ROR on their own behalf, and perform normal IPv6 ND operations the
same as for any node.) When a Proxy/Server acting as an ROR receives
an authentic NS message used for route optimization, it searches for

a neighbor cache entry for the target Client. The ROR then returns a
solicited NA message while creating a neighbor cache entry for the
ROS, and also creates or updates a target Client "Report List" entry
for the ROS and sets a "ReportTime" variable for the entry to
REPORT_TIME seconds. The ROR resets ReportTime when it receives a
new authentic NS message, and otherwise decrements ReportTime while
no authentic NS messages have been received. It is RECOMMENDED that
REPORT_TIME be set to the default constant value REACHABLE_TIME plus
10 seconds (40 seconds by default) to allow a window for route
optimization to converge before ReportTime decrements below
REACHABLE_TIME.

When the ROS receives a solicited NA message response to its NS
message used for route optimization, it creates or updates a neighbor
cache entry for the target network-layer and link-layer addresses.
The ROS then (re)sets ReachableTime for the neighbor cache entry to
REACHABLE_TIME seconds and uses this value to determine whether
carrier packets can be forwarded directly to the target, i.e.,
instead of via a default route. The ROS otherwise decrements
ReachableTime while no further solicited NA messages arrive. It is
RECOMMENDED that REACHABLE_TIME be set to the default constant value
30 seconds as specified in [RFC4861].

AERO nodes also use the value MAX_UNICAST_SOLICIT to limit the number
of NS keepalives sent when a correspondent may have gone unreachable,
the value MAX_RTR_SOLICITATIONS to limit the number of RS messages
sent without receiving an RA and the value MAX_NEIGHBOR_ADVERTISEMENT
to limit the number of unsolicited NAs that can be sent based on a
single event. It is RECOMMENDED that MAX_UNICAST_SOLICIT,
MAX_RTR_SOLICITATIONS and MAX_NEIGHBOR_ADVERTISEMENT be set to 3 the
same as specified in [RFC4861].

Different values for DEPART_TIME, REPORT_TIME, REACHABLE_TIME,
MAX_UNICAST_SOLICIT, MAX_RTR_SOLCITATIONS and
MAX_NEIGHBOR_ADVERTISEMENT MAY be administratively set; however, if
different values are chosen, all nodes on the link MUST consistently
configure the same values. Most importantly, DEPART_TIME and
REPORT_TIME SHOULD be set to a value that is sufficiently longer than
REACHABLE_TIME to avoid packet loss due to stale route optimization
state.

3.5.1. OMNI Neighbor Interface Attributes

OMNI interface IPv6 ND messages include OMNI options
[I-D.templin-6man-omni-interface] with Interface Attributes that
provide Link-Layer Address and QoS Preference information for the
neighbor's underlying interfaces. This information is stored in the
neighbor cache and provides the basis for the forwarding algorithm

specified in Section 3.10. The information is cumulative and
reflects the union of the OMNI information from the most recent ND
messages received from the neighbor; it is therefore not required
that each ND message contain all neighbor information.

The OMNI option Interface Attributes for each underlying interface
includes a two-part "Link-Layer Address" consisting of a simple IP
encapsulation address determined by the FMT and L2ADDR fields and an
ADM-ULA determined by the SRT and LHS fields. Underlying interfaces
are further selected based on their associated preference values
"high", "medium", "low" or "disabled".

Note: the OMNI option is distinct from any Source/Target Link-Layer
Address Options (S/TLLAOs) that may appear in an ND message according
to the appropriate IPv6 over specific link layer specification (e.g.,
[RFC2464]). If both an OMNI option and S/TLLAO appear, the former
pertains to encapsulation addresses while the latter pertains to the
native L2 address format of the underlying media.

3.5.2. OMNI Neighbor Advertisement Message Flags

As discussed in Section 4.4 of [RFC4861] NA messages include three
flag bits R, S and O. OMNI interface NA messages treat the flags as
follows:

o R: The R ("Router") flag is set to 1 in the NA messages sent by
   all AERO/OMNI node types. Simple hosts that would set R to 0 do
   not occur on the OMNI link itself, but may occur on the downstream
   links of Clients and Relays.

o S: The S ("Solicited") flag is set exactly as specified in
   Section 4.4. of [RFC4861], i.e., it is set to 1 for Solicited NAs
   and set to 0 for Unsolicited NAs (both unicast and multicast).

o O: The O ("Override") flag is set to 0 for solicited proxy NAs and
   set to 1 for all other solicited and unsolicited NAs. For further
   study is whether solicited NAs for anycast targets apply for OMNI
   links. Since MNP-LLAs must be uniquely assigned to Clients to
   support correct ND protocol operation, however, no role is
   currently seen for assigning the same MNP-LLA to multiple Clients.

3.6. OMNI Interface Encapsulation and Re-encapsulation

The OMNI interface admits original IP packets then (acting as an OAL
source) performs OAL encapsulation and fragmentation as specified in
[I-D.templin-6man-omni-interface] while including an ORH if necessary
as specified in Section 3.2.4. OAL encapsulation produces OAL
packets, while OAL fragmentation turns them into OAL fragments which

are then encapsulated in *NET headers as carrier packets. For
carrier packets undergoing re-encapsulation at an intermediate node,
the OMNI interface removes the *NET encapsulation headers from the
first segment and re-encapsulates the packet in new *NET
encapsulation headers for the next segment.

When a Proxy/Server or Relay re-encapsulates a carrier packet
received from a Client that includes an OAL but no ORH, it inserts an
ORH immediately following the OAL header and adjusts the OAL payload
length and destination address field. The inserted ORH will be
removed by the final-hop Bridge, but its insertion and removal will
not interfere with reassembly at the final destination. For this
reason, Clients must reserve 40 bytes for a maximum-length ORH when
they perform OAL encapsulation (see: Section 3.9).

3.7. OMNI Interface Decapsulation

OMNI interfaces (acting as OAL destinations) decapsulate and
reassemble OAL packets into original IP packets destined either to
the AERO node itself or to a destination reached via an interface
other than the OMNI interface the original IP packet was received on.
When carrier packets containing OAL fragments arrive, the OMNI
interface reassembles as discussed in Section 3.9.

3.8. OMNI Interface Data Origin Authentication

AERO nodes employ simple data origin authentication procedures. In
particular:

o AERO Bridges and Proxy/Servers accept carrier packets (including
   either data or control messages) received from the (secured)
   spanning tree.

o AERO Proxy/Servers and Clients accept carrier packets and original
   IP packets that originate from within the same secured ANET.

o AERO Clients and Relays accept original IP packets from downstream
   network correspondents based on ingress filtering.

o AERO Clients, Relays and Proxy/Servers verify carrier packet UDP/
   IP encapsulation addresses according to [RFC4380].

o AERO Clients (as well as Proxy/Servers and Relays when acting as
   OAL destinations) accept OAL packets/fragments with Identification
   values within the current window for the OAL source.

AERO nodes silently drop any packets that do not satisfy the above
data origin authentication procedures. Further security
considerations are discussed in Section 6.

3.9. OMNI Interface 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 employs an OMNI Adaptation Layer (OAL) that
accommodates multiple underlying links with diverse MTUs while
observing both a minimum and per-path Maximum Payload Size (MPS).
The functions of the OAL and the OMNI interface MTU/MRU/MPS are
specified in Section 5 of [I-D.templin-6man-omni-interface] with MTU/
MRU both set to the constant value 9180 bytes, with minimum MPS set
to 400 bytes, and with path MPS set to a potentially larger value
depending on the underlying path.

When the network layer presents an original IP packet to the OMNI
interface, the OAL source encapsulates and fragments the original IP
packet if necessary. When the network layer presents the OMNI
interface with multiple original IP packets bound to the same OAL
destination, the OAL source can concatenate them together into a
single OAL super-packet as discussed in
[I-D.templin-6man-omni-interface]. The OAL source then fragments the
OAL packet if necessary according to the minimum/path MPS such that
the OAL headers appear in each fragment while the original IP packet
header appears only in the first fragment. The OAL source then
encapsulates each OAL fragment in *NET headers for transmission as
carrier packets over an underlying interface connected to either a
physical link such as Ethernet, WiFi and the like or a virtual link
such as an Internet or higher-layer tunnel (see the definition of
link in [RFC8200]).

Note: A Client that does not (yet) have neighbor cache state for a
target may omit the ORH in carrier packets with the understanding
that a Proxy/Server may insert an ORH on its behalf. For this
reason, Clients reserve 40 bytes for the largest possible ORH in
their OAL fragment size calculations.

Note: Although the ORH may be removed by a Bridge on the path (see:
Section 3.10.3), this does not interfere with the destination's
ability to reassemble. This is due to the fact that the ORH is not
included in the fragmentable part; therefore, its removal does not
invalidate the offset values in any fragment headers.

3.10. OMNI Interface Forwarding Algorithm

   Original IP packets enter a node's OMNI interface either from the
   network layer (i.e., from a local application or the IP forwarding
   system) while carrier packets enter from the link layer (i.e., from
   an OMNI interface neighbor). All original IP packets and carrier
   packets entering a node's OMNI interface first undergo data origin
   authentication as discussed in Section 3.8. Those that satisfy data
   origin authentication are processed further, while all others are
   dropped silently.

   Original IP packets that enter the OMNI interface from the network
   layer are forwarded to an OMNI interface neighbor using OAL
   encapsulation and fragmentation to produce carrier packets for
   transmission over underlying interfaces. (If routing indicates that
   the original IP packet should instead be forwarded back to the
   network layer, the packet is dropped to avoid looping). Carrier
   packets that enter the OMNI interface from the link layer are either
   re-encapsulated and re-admitted into the OMNI link, or reassembled
   and forwarded to the network layer where they are subject to either
   local delivery or IP forwarding. In all cases, the OAL MUST NOT
   decrement the network layer TTL/Hop-count since its forwarding
   actions occur below the network layer.

   OMNI interfaces may have multiple underlying interfaces and/or
   neighbor cache entries for neighbors with multiple underlying
   interfaces (see Section 3.3). The OAL uses interface attributes and/
   or traffic classifiers (e.g., DSCP value, port number, flow
   specification, etc.) to select an outgoing underlying interface for
   each OAL packet based on the node's own QoS preferences, and also to
   select a destination link-layer address based on the neighbor's
   underlying interface with the highest preference. AERO
   implementations SHOULD allow for QoS preference values to be modified
   at runtime through network management.

   If multiple outgoing interfaces and/or neighbor interfaces have a
   preference of "high", the AERO node replicates the OAL packet and
   sends one copy via each of the (outgoing / neighbor) interface pairs;
   otherwise, the node sends a single copy of the OAL packet via an
   interface with the highest preference. (While not strictly required,
   successful delivery may be more likely when all OAL fragments of the
   same OAL packet are sent over the same underlying interface.) AERO
   nodes keep track of which underlying interfaces are currently
   "reachable" or "unreachable", and only use "reachable" interfaces for
   forwarding purposes.

   The following sections discuss the OMNI interface forwarding
   algorithms for Clients, Proxy/Servers and Bridges. In the following

discussion, an original IP packet's destination address is said to
"match" if it is the same as a cached address, or if it is covered by
a cached prefix (which may be encoded in an MNP-LLA).

3.10.1. Client Forwarding Algorithm

When an original IP packet enters a Client's OMNI interface from the
network layer the Client searches for a neighbor cache entry that
matches the destination. If there is a match, the Client selects one
or more "reachable" neighbor interfaces in the entry for forwarding
purposes. If there is no neighbor cache entry, the Client instead
forwards the original IP packet toward a Proxy/Server. The Client
(acting as an OAL source) performs OAL encapsulation and sets the OAL
destination address to the MNP-ULA if there is a matching neighbor
cache entry; otherwise, it sets the OAL destination to the ADM-ULA of
the Proxy/Server. If the Client has multiple original IP packets to
send to the same neighbor, it can concatenate them in a single super-
packet [I-D.templin-6man-omni-interface]. The OAL source then
performs fragmentation to create OAL fragments (see: Section 3.9),
appends any *NET encapsulation, and sends the resulting carrier
packets over underlying interfaces to the neighbor acting as an OAL
destination.

If the neighbor interface selected for forwarding is located on the
same OMNI link segment and not behind a NAT, the Client forwards the
carrier packets directly according to the L2ADDR information for the
neighbor. If the neighbor interface is behind a NAT on the same OMNI
link segment, the Client instead forwards the initial carrier packets
to its Proxy/Server and initiates NAT traversal procedures. If the
Client's intended source underlying interface is also behind a NAT
and located on the same OMNI link segment, it sends a "direct bubble"
over the interface per [RFC6081][RFC4380] to the L2ADDR found in the
neighbor cache in order to establish state in its own NAT by
generating traffic toward the neighbor (note that no response to the
bubble is expected).

The Client next sends an NS(NUD) message toward the MNP-ULA of the
neighbor via its Proxy/Server as discussed in Section 3.15. If the
Client receives an NA(NUD) from the neighbor over the underlying
interface, it marks the neighbor interface as "trusted" and sends
future carrier packets directly to the L2ADDR information for the
neighbor instead of indirectly via the Proxy/Server. The Client must
honor the neighbor cache maintenance procedure by sending additional
direct bubbles and/or NS/NA(NUD) messages as discussed in
[RFC6081][RFC4380] in order to keep NAT state alive as long as
carrier packets are still flowing.

When an carrier packet enters a Client's OMNI interface from the
link-layer, if the OAL destination matches one of the Client's MNPs
or LLAs the Client (acting as an OAL destination) reassembles and
decapsulates as necessary and delivers the original IP packet to the
network layer. Otherwise, the Client drops the original IP packet
and MAY return a network-layer ICMP Destination Unreachable message
subject to rate limiting (see: Section 3.11).

3.10.2. Proxy/Server Forwarding Algorithm

For IPv6 ND control messages originating from or destined to a
Client, the Proxy/Server intercepts the message and updates its
neighbor cache entry for the Client. If acting in Proxy-only mode,
it then forwards a (proxied) copy of the control message to one or
more neighbors. (For example, the Proxy forwards a proxied version
of a Client's NS/RS message to the neighbor, and forwards a proxied
version of the neighbor's NA/RA reply to the Client.)

When the Proxy/Server receives an original IP packet from the network
layer, it drops the packet if routing indicates that it should be
forwarded back to the network layer to avoid looping. Otherwise, the
Proxy/Server regards the original IP packet the same as if it had
arrived as carrier packets with OAL destination set to its own ADM-
ULA. When the Proxy/Server receives carrier packets with OAL
destination set to its own ADM-ULA, it performs OAL reassembly if
necessary to obtain the original IP packet.

The Proxy/Server next searches for a neighbor cache entry that
matches the original IP destination and proceeds as follows:

o if the original IP packet destination matches a neighbor cache
   entry, the Proxy/Sever uses one or more "reachable" neighbor
   interfaces in the entry for packet forwarding using OAL
   encapsulation and fragmentation according to the cached link-layer
   address information. If the neighbor interface is in a different
   OMNI link segment, the Proxy/Server forwards the resulting carrier
   packets to a Bridge; otherwise, it forwards the carrier packets
   directly to the neighbor. If the neighbor is behind a NAT, the
   Proxy/Server instead forwards initial carrier packets via a Bridge
   while sending an NS(NUD) to the neighbor. When the Proxy/Server
   receives the NA(NUD), it can begin forwarding carrier packets
   directly to the neighbor the same as discussed in Section 3.10.1
   while sending additional NS(NUD) messages as necessary to maintain
   NAT state. Note that no direct bubbles are necessary since the
   Proxy/Server is by definition not located behind a NAT.

o else, if the original IP destination matches a non-MNP route in
   the IP forwarding table or an ADM-LLA assigned to the Proxy/

Server's OMNI interface, the Proxy/Server presents the original IP
packet to the network layer for local delivery or IP forwarding.

o else, the Proxy/Server initiates address resolution as discussed
   in Section 3.14, while retaining initial original IP packets in a
   small queue awaiting address resolution completion.

When the Proxy/Server receives a carrier packet with OAL destination
set to a non-MNP ULA, it accepts the carrier packet only if data
origin authentication succeeds and if there is a network layer
routing table for a GUA route that matches the non-MNP ULA. If there
is no route, the Proxy/Server drops the carrier packet; otherwise, it
reassembles and decapsulates to obtain the original IP packet and
presents it to the network layer where it will be delivered according
to standard IP forwarding.

When the Proxy/Server receives a carrier packet with OAL destination
set to an MNP-ULA, it accepts the carrier packet only if data origin
authentication succeeds and if there is a neighbor cache entry that
matches the OAL destination. If the neighbor cache entry state is
DEPARTED, the Proxy/Server inserts an ORH that encodes the MNP-ULA
destination suffix and changes the OAL destination address to the
ADM-ULA of the new Proxy/Server, then re-encapsulates the carrier
packet and forwards it to a Bridge which will eventually deliver it
to the new Proxy/Server.

If the neighbor cache state is REACHABLE, the Proxy/Server can
instead either reassemble first and then re-encapsulate/re-fragment
before forwarding to the Client or forward the raw fragments on to
the Client which then must reassemble. In the former case, the
Proxy/Server can re-fragment to a size that better matches the link
MTU for the Client, which may be important for low-end links with
large MTUs. In the latter case, the Client may receive fragments
that are smaller than its link MTU but can still be reassembled; this
case may provide an important performance benefit to Proxy/Servers by
permitting them to avoid excessive reassembly and re-fragmentation
overhead. In either case, the Proxy/Server can return a PTB if
necessary (see: [I-D.templin-6man-omni-interface]) when it receives a
carrier packet containing an OAL first fragment.

Note: If the Proxy/Server has multiple original IP packets to send to
the same neighbor, it can concatenate them in a single OAL super-
packet [I-D.templin-6man-omni-interface].

3.10.3. **Bridge Forwarding Algorithm**

   Bridges forward OAL fragments over secured tunnels the same as any
   IPv6 router. When the Bridge receives a carrier packet or an
   original IP packet via a secured tunnel, it removes the outer *NET
   header and searches for a forwarding table entry that matches the OAL
   destination address. The Bridge then processes the packet as
   follows:

   o if the packet is a carrier packet with a destination that matches
      its ADM-ULA Subnet Router Anycast address the Bridge processes the
      carrier packet locally before forwarding. The Bridge drops the
      carrier packet if it does not include an ORH; otherwise, for
      NA(NUD) messages the Bridge replaces the OMNI option Interface
      Attributes sub-option with information for its own interface while
      retaining the ifIndex value supplied by the NA(NUD) message
      source. The Bridge next examines the ORH FMT code. If the code
      indicates the destination is a Client on the open *NET (or, a
      Client behind a NAT for which NAT traversal procedures have
      already converged) the Bridge removes the ORH then writes the MNP-
      ULA formed from the ORH Destination Suffix into the OAL
      destination. The Bridge then re-encapsulates the carrier packet
      and forwards it to the ORH L2ADDR. For all other destination
      cases, the Bridge instead writes the ADM-ULA formed from the ORH
      SRT/LHS into the OAL destination address and forwards the carrier
      packet to the ADM-ULA Proxy/Server while invoking NAT traversal
      procedures the same as for Proxy/Servers if necessary, noting that
      no direct bubbles are necessary since only the target Client and
      not the Bridge is behind a NAT.

   o else, if the packet is a carrier packet with a destination that
      matches a forwarding table entry the Bridge forwards the carrier
      packet via a secured tunnel to the next hop. If the destination
      matches an MSP without matching an MNP, however, the Bridge
      instead drops the packet and returns an ICMP Destination
      Unreachable message subject to rate limiting (see: Section 3.11).

   o else, if the packet is an original IP packet with a destination
      that matches one of the Bridge's own addresses, the Bridge submits
      the original IP packet for local delivery to support local
      applications such as routing protocols.

   o else, the Bridge drops the packet and returns an ICMP Destination
      Unreachable as above.

   As for any IP router, the Bridge decrements the TTL/Hop Limit when it
   forwards the carrier packet. Therefore, when an OAL header is
   present only the Hop Limit in the OAL header is decremented and not

the TTL/Hop Limit in the original IP packet header. Bridges do not
insert OAL/ORH headers themselves; instead, they act as IPv6 routers
and forward carrier packets based on their destination addresses.

3.11. OMNI Interface Error Handling

When an AERO node admits an original IP packet into the OMNI
interface, it may receive link-layer or network-layer error
indications.

A link-layer error indication is an ICMP error message generated by a
router in the INET on the path to the neighbor or by the neighbor
itself. The message includes an IP header with the address of the
node that generated the error as the source address and with the
link-layer address of the AERO node as the destination address.

The IP header is followed by an ICMP header that includes an error
Type, Code and Checksum. Valid type values include "Destination
Unreachable", "Time Exceeded" and "Parameter Problem"
[RFC0792][RFC4443]. (OMNI interfaces ignore all link-layer IPv4
"Fragmentation Needed" and IPv6 "Packet Too Big" messages since they
only emit carrier packets that are guaranteed to be no larger than
the IP minimum link MTU as discussed in Section 3.9.)

The ICMP header is followed by the leading portion of the original IP
packet that generated the error, also known as the "packet-in-error".
For ICMPv6, [RFC4443] specifies that the packet-in-error includes:
"As much of invoking packet as possible without the ICMPv6 packet
exceeding the minimum IPv6 MTU" (i.e., no more than 1280 bytes). For
ICMPv4, [RFC0792] specifies that the packet-in-error includes:
"Internet Header + 64 bits of Original Data Datagram", however
[RFC1812] Section 4.3.2.3 updates this specification by stating: "the
ICMP datagram SHOULD contain as much of the original datagram as
possible without the length of the ICMP datagram exceeding 576
bytes".

The link-layer error message format is shown in Figure 5 (where, "L2"
and "L3" refer to link-layer and network-layer, respectively):

```
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ~                               ~
        |         L2 IP Header of       |
        |          error message        |
        ~                               ~
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |         L2 ICMP Header        |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
        ~                               ~   P
        |    IP and other encapsulation |   a
        |  headers of original IP packet|   c
        ~                               ~   k
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   e
        ~                               ~   t
        |          IP header of         |
        |        original IP packet     |   i
        ~                               ~   n
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ~                               ~   e
        |     Upper layer headers and   |   r
        |      leading portion of body  |   r
        |    of the original IP packet  |   o
        ~                               ~   r
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
```
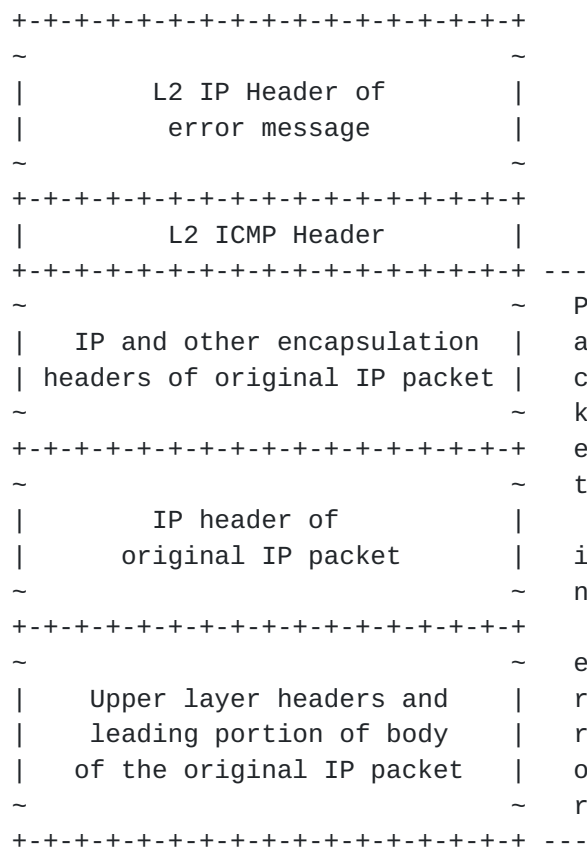
        Figure 5: OMNI Interface Link-Layer Error Message Format

   The AERO node rules for processing these link-layer error messages
   are as follows:

   o When an AERO node receives a link-layer Parameter Problem message,
      it processes the message the same as described as for ordinary
      ICMP errors in the normative references [RFC0792][RFC4443].

   o When an AERO node receives persistent link-layer Time Exceeded
      messages, the IP ID field may be wrapping before earlier fragments
      awaiting reassembly have been processed. In that case, the node
      should begin including integrity checks and/or institute rate
      limits for subsequent packets.

   o When an AERO node receives persistent link-layer Destination
      Unreachable messages in response to carrier packets that it sends
      to one of its neighbor correspondents, the node should process the
      message as an indication that a path may be failing, and
      optionally initiate NUD over that path. If it receives
      Destination Unreachable messages over multiple paths, the node
      should allow future carrier packets destined to the correspondent

to flow through a default route and re-initiate route
optimization.

o When an AERO Client receives persistent link-layer Destination
   Unreachable messages in response to carrier packets that it sends
   to one of its neighbor Proxy/Servers, the Client should mark the
   path as unusable and use another path. If it receives Destination
   Unreachable messages on many or all paths, the Client should
   associate with a new Proxy/Server and release its association with
   the old Proxy/Server as specified in Section 3.16.5.

o When an AERO Proxy/Server receives persistent link-layer
   Destination Unreachable messages in response to carrier packets
   that it sends to one of its neighbor Clients, the Proxy/Server
   should mark the underlying path as unusable and use another
   underlying path.

o When an AERO Proxy/Server receives link-layer Destination
   Unreachable messages in response to a carrier packet that it sends
   to one of its permanent neighbors, it treats the messages as an
   indication that the path to the neighbor may be failing. However,
   the dynamic routing protocol should soon reconverge and correct
   the temporary outage.

When an AERO Bridge receives a carrier packet for which the network-
layer destination address is covered by an MSP, the Bridge drops the
packet if there is no more-specific routing information for the
destination and returns a network-layer Destination Unreachable
message subject to rate limiting. The Bridge writes the network-
layer source address of the original IP packet as the destination
address and uses one of its non link-local addresses as the source
address of the message.

When an AERO node receives a carrier packet for which the reassembly
buffer it too small, it drops the packet and returns a network-layer
Packet Too Big (PTB) message. The node first writes the MRU value
into the PTB message MTU field, writes the network-layer source
address of the original IP packet as the destination address and
writes one of its non link-local addresses as the source address.

3.12. AERO Router Discovery, Prefix Delegation and Autoconfiguration

AERO Router Discovery, Prefix Delegation and Autoconfiguration are
coordinated as discussed in the following Sections.

3.12.1. AERO Service Model

   Each AERO Proxy/Server on the OMNI link is configured to facilitate
   Client prefix delegation/registration requests. Each Proxy/Server is
   provisioned with a database of MNP-to-Client ID mappings for all
   Clients enrolled in the AERO service, as well as any information
   necessary to authenticate each Client. The Client database is
   maintained by a central administrative authority for the OMNI link
   and securely distributed to all Proxy/Servers, e.g., via the
   Lightweight Directory Access Protocol (LDAP) [RFC4511], via static
   configuration, etc. Clients receive the same service regardless of
   the Proxy/Servers they select.

   AERO Clients and Proxy/Servers use ND messages to maintain neighbor
   cache entries. AERO Proxy/Servers configure their OMNI interfaces as
   advertising NBMA interfaces, and therefore send unicast RA messages
   with a short Router Lifetime value (e.g., ReachableTime seconds) in
   response to a Client's RS message. Thereafter, Clients send
   additional RS messages to keep Proxy/Server state alive.

   AERO Clients and Proxy/Servers include prefix delegation and/or
   registration parameters in RS/RA messages (see
   [I-D.templin-6man-omni-interface]). The ND messages are exchanged
   between Client and Proxy/Server according to the prefix management
   schedule required by the service. If the Client knows its MNP in
   advance, it can employ prefix registration by including its MNP-LLA
   as the source address of an RS message and with an OMNI option with
   valid prefix registration information for the MNP. If the Proxy/
   Server accepts the Client's MNP assertion, it injects the MNP into
   the routing system and establishes the necessary neighbor cache
   state. If the Client does not have a pre-assigned MNP, it can
   instead employ prefix delegation by including the unspecified address
   (::) as the source address of an RS message and with an OMNI option
   with prefix delegation parameters to request an MNP.

   The following sections specify the Client and Proxy/Server behavior.

3.12.2. AERO Client Behavior

   AERO Clients discover the addresses of Proxy/Servers in a similar
   manner as described in [RFC5214]. Discovery methods include static
   configuration (e.g., from a flat-file map of Proxy/Server addresses
   and locations), or through an automated means such as Domain Name
   System (DNS) name resolution [RFC1035]. Alternatively, the Client
   can discover Proxy/Server addresses through a layer 2 data link login
   exchange, or through a unicast RA response to a multicast/anycast RS
   as described below. In the absence of other information, the Client
   can resolve the DNS Fully-Qualified Domain Name (FQDN)

"linkupnetworks.[domainname]" where "linkupnetworks" is a constant
text string and "[domainname]" is a DNS suffix for the OMNI link
(e.g., "example.com").

To associate with a Proxy/Server, the Client acts as a requesting
router to request MNPs. The Client prepares an RS message with
prefix management parameters and includes a Nonce and Timestamp
option if the Client needs to correlate RA replies. If the Client
already knows the Proxy/Server's ADM-LLA, it includes the LLA as the
network-layer destination address; otherwise, the Client includes the
(link-local) All-Routers multicast as the network-layer destination.
If the Client already knows its own MNP-LLA, it can use the MNP-LLA
as the network-layer source address and include an OMNI option with
prefix registration information. Otherwise, the Client uses the
unspecified address (::) as the network-layer source address and
includes prefix delegation parameters in the OMNI option (see:
[I-D.templin-6man-omni-interface]). The Client includes Interface
Attributes corresponding to the underlying interface over which it
will send the RS message, and MAY include additional Interface
Attributes specific to other underlying interfaces.

The Client then sends the RS message (either directly via Direct
interfaces, via a VPN for VPNed interfaces, via a Proxy for ANET
interfaces or via INET encapsulation for INET interfaces) while using
OAL encapsulation/fragmentation, then waits for an RA message reply
(see Section 3.12.3). The Client retries up to MAX_RTR_SOLICITATIONS
times until an RA is received. If the Client receives no RAs, or if
it receives an RA with Router Lifetime set to 0, the Client SHOULD
abandon this Proxy/Server and try another Proxy/Server. Otherwise,
the Client processes the prefix information found in the RA message.

When the Client processes an RA, it first performs OAL reassembly and
decapsulation then creates a neighbor cache entry with the Proxy/
Server's ADM-LLA as the network-layer address and the Proxy/Server's
encapsulation and/or link-layer addresses as the link-layer address.
The Client next records the RA Router Lifetime field value in the
neighbor cache entry as the time for which the Proxy/Server has
committed to maintaining the MNP in the routing system via this
underlying interface, and caches the other RA configuration
information including Cur Hop Limit, M and O flags, Reachable Time
and Retrans Timer. The Client then autoconfigures MNP-LLAs for any
delegated MNPs and assigns them to the OMNI interface. The Client
also caches any MSPs included in Route Information Options (RIOs)
[RFC4191] as MSPs to associate with the OMNI link, and assigns the
MTU value in the MTU option to the underlying interface.

The Client then registers additional underlying interfaces with the
Proxy/Server by sending RS messages via each additional interface as

described above. The RS messages include the same parameters as for
the initial RS/RA exchange, but with destination address set to the
Proxy/Server's ADM-LLA. The Client finally sub-delegates the MNPs to
its attached EUNs and/or the Client's own internal virtual interfaces
as described in [I-D.templin-v6ops-pdhost] to support the Client's
downstream attached "Internet of Things (IoT)". The Client then
sends additional RS messages over each underlying interface before
the Router Lifetime received for that interface expires.

After the Client registers its underlying interfaces, it may wish to
change one or more registrations, e.g., if an interface changes
address or becomes unavailable, if QoS preferences change, etc. To
do so, the Client prepares an RS message to send over any available
underlying interface as above. The RS includes an OMNI option with
prefix registration/delegation information, with Interface Attributes
specific to the selected underlying interface, and with any
additional Interface Attributes specific to other underlying
interfaces. When the Client receives the Proxy/Server's RA response,
it has assurance that the Proxy/Server has been updated with the new
information.

If the Client wishes to discontinue use of a Proxy/Server it issues
an RS message over any underlying interface with an OMNI option with
a prefix release indication. When the Proxy/Server processes the
message, it releases the MNP, sets the neighbor cache entry state for
the Client to DEPARTED and returns an RA reply with Router Lifetime
set to 0. After a short delay (e.g., 2 seconds), the Proxy/Server
withdraws the MNP from the routing system.

3.12.3. AERO Proxy/Server Behavior

AERO Proxy/Servers act as IP routers and support a prefix delegation/
registration service for Clients. Proxy/Servers arrange to add their
ADM-LLAs to a static map of Proxy/Server addresses for the link and/
or the DNS resource records for the FQDN
"linkupnetworks.[domainname]" before entering service. Proxy/Server
addresses should be geographically and/or topologically referenced,
and made available for discovery by Clients on the OMNI link.

When a Proxy/Server receives a prospective Client's RS message on its
OMNI interface, it SHOULD return an immediate RA reply with Router
Lifetime set to 0 if it is currently too busy or otherwise unable to
service the Client. Otherwise, the Proxy/Server performs OAL
reassembly and decapsulation, then authenticates the RS message and
processes the prefix delegation/registration parameters. The Proxy/
Server first determines the correct MNPs to provide to the Client by
processing the MNP-LLA prefix parameters and/or the DHCPv6 OMNI sub-
option. When the Proxy/Server returns the MNPs, it also creates a

forwarding table entry for the MNP-ULA corresponding to each MNP so
that the MNPs are propagated into the routing system (see:
Section 3.2.3). For IPv6, the Proxy/Server creates an IPv6
forwarding table entry for each MNP. For IPv4, the Proxy/Server
creates an IPv6 forwarding table entry with the IPv4-compatibility
MNP-ULA prefix corresponding to the IPv4 address.

The Proxy/Server next creates a neighbor cache entry for the Client
using the base MNP-LLA as the network-layer address and with lifetime
set to no more than the smallest prefix lifetime. Next, the Proxy/
Server updates the neighbor cache entry by recording the information
in each Interface Attributes sub-option in the RS OMNI option. The
Proxy/Server also records the actual OAL/INET addresses in the
neighbor cache entry.

Next, the Proxy/Server prepares an RA message using its ADM-LLA as
the network-layer source address and the network-layer source address
of the RS message as the network-layer destination address. The
Proxy/Server sets the Router Lifetime to the time for which it will
maintain both this underlying interface individually and the neighbor
cache entry as a whole. The Proxy/Server also sets Cur Hop Limit, M
and O flags, Reachable Time and Retrans Timer to values appropriate
for the OMNI link. The Proxy/Server includes the MNPs, any other
prefix management parameters and an OMNI option with no Interface
Attributes. The Proxy/Server then includes one or more RIOs that
encode the MSPs for the OMNI link, plus an MTU option (see
Section 3.9). The Proxy/Server finally forwards the message to the
Client using OAL encapsulation/fragmentation as necessary.

After the initial RS/RA exchange, the Proxy/Server maintains a
ReachableTime timer for each of the Client's underlying interfaces
individually (and for the Client's neighbor cache entry collectively)
set to expire after ReachableTime seconds. If the Client (or Proxy)
issues additional RS messages, the Proxy/Server sends an RA response
and resets ReachableTime. If the Proxy/Server receives an ND message
with a prefix release indication it sets the Client's neighbor cache
entry to the DEPARTED state and withdraws the MNP from the routing
system after a short delay (e.g., 2 seconds). If ReachableTime
expires before a new RS is received on an individual underlying
interface, the Proxy/Server marks the interface as DOWN. If
ReachableTime expires before any new RS is received on any individual
underlying interface, the Proxy/Server sets the neighbor cache entry
state to STALE and sets a 10 second timer. If the Proxy/Server has
not received a new RS or ND message with a prefix release indication
before the 10 second timer expires, it deletes the neighbor cache
entry and withdraws the MNP from the routing system.

   The Proxy/Server processes any ND messages pertaining to the Client
   and returns an NA/RA reply in response to solicitations. The Proxy/
   Server may also issue unsolicited RA messages, e.g., with reconfigure
   parameters to cause the Client to renegotiate its prefix delegation/
   registrations, with Router Lifetime set to 0 if it can no longer
   service this Client, etc. Finally, If the neighbor cache entry is in
   the DEPARTED state, the Proxy/Server deletes the entry after
   DepartTime expires.

   Note: Clients SHOULD notify former Proxy/Servers of their departures,
   but Proxy/Servers are responsible for expiring neighbor cache entries
   and withdrawing routes even if no departure notification is received
   (e.g., if the Client leaves the network unexpectedly). Proxy/Servers
   SHOULD therefore set Router Lifetime to ReachableTime seconds in
   solicited RA messages to minimize persistent stale cache information
   in the absence of Client departure notifications. A short Router
   Lifetime also ensures that proactive RS/RA messaging between Clients
   and Proxy/Servers will keep any NAT state alive (see above).

   Note: All Proxy/Servers on an OMNI link MUST advertise consistent
   values in the RA Cur Hop Limit, M and O flags, Reachable Time and
   Retrans Timer fields the same as for any link, since unpredictable
   behavior could result if different Proxy/Servers on the same link
   advertised different values.

3.12.3.1. DHCPv6-Based Prefix Registration

   When a Client is not pre-provisioned with an MNP-LLA, it will need
   for the Proxy/Server to select one or more MNPs on its behalf and set
   up the correct state in the AERO routing service. (A Client with a
   pre-provisioned MNP may also request the Proxy/Server to select
   additional MNPs.) The DHCPv6 service [RFC8415] is used to support
   this requirement.

   When a Client needs to have the Proxy/Server select MNPs, it sends an
   RS message with source address set to the unspecified address (::)
   and with an OMNI option that includes a DHCPv6 message sub-option
   with DHCPv6 Prefix Delegation (DHCPv6-PD) parameters. When the
   Proxy/Server receives the RS message, it extracts the DHCPv6-PD
   message from the OMNI option.

   The Proxy/Server then acts as a "Proxy DHCPv6 Client" in a message
   exchange with the locally-resident DHCPv6 server, which delegates
   MNPs and returns a DHCPv6-PD Reply message. (If the Proxy/Server
   wishes to defer creation of MN state until the DHCPv6-PD Reply is
   received, it can instead act as a Lightweight DHCPv6 Relay Agent per
   [RFC6221] by encapsulating the DHCPv6-PD message in a Relay-forward/
   reply exchange with Relay Message and Interface ID options.)

   When the Proxy/Server receives the DHCPv6-PD Reply, it adds a route
   to the routing system and creates an MNP-LLA based on the delegated
   MNP. The Proxy/Server then sends an RA back to the Client with the
   (newly-created) MNP-LLA as the destination address and with the
   DHCPv6-PD Reply message coded in the OMNI option. When the Client
   receives the RA, it creates a default route, assigns the Subnet
   Router Anycast address and sets its MNP-LLA based on the delegated
   MNP.

   Note: See [I-D.templin-6man-omni-interface] for an MNP delegation
   alternative in which the Client can optionally avoid including a
   DHCPv6 message sub-option. Namely, when the Client requests a single
   MNP it can set the RS source to the unspecified address (::) and
   include a Node Identification sub-option and Preflen in the OMNI
   option (but with no DHCPv6 message sub-option). When the Proxy/
   Server receives the RS message, it forwards a self-generated DHCPv6
   Solicit message to the DHCPv6 server on behalf of the Client. When
   the Proxy/Server receives the DHCPv6 Reply, it prepares an RA message
   with an OMNI option with Preflen information (but with no DHCPv6
   message sub-option), then places the (newly-created) MNP-LLA in the
   RA destination address and returns the message to the Client.

3.13. The AERO Proxy

   Clients may connect to protected-spectrum ANETs that employ physical
   and/or link-layer security services to facilitate communications to
   Proxy/Servers in outside INETs. In that case, the ANET can employ an
   AERO Proxy. The Proxy is located at the ANET/INET border and listens
   for RS messages originating from or RA messages destined to ANET
   Clients. The Proxy acts on these control messages as follows:

   o when the Proxy receives an RS message from a new ANET Client, it
      first authenticates the message then examines the network-layer
      destination address. If the destination address is a Proxy/
      Server's ADM-LLA, the Proxy proceeds to the next step. Otherwise,
      if the destination is (link-local) All-Routers multicast, the
      Proxy selects a "nearby" Proxy/Server that is likely to be a good
      candidate to serve the Client and replaces the destination address
      with the Proxy/Server's ADM-LLA. Next, the Proxy creates a proxy
      neighbor cache entry and caches the Client and Proxy/Server link-
      layer addresses along with the OMNI option information and any
      other identifying information including Transaction IDs, Client
      Identifiers, Nonce values, etc. The Proxy finally encapsulates
      the (proxyed) RS message in an OAL header with source set to the
      Proxy's ADM-ULA and destination set to the Proxy/Server's ADM-ULA.
      The Proxy also includes an OMNI header with an Interface
      Attributes option that includes its own INET address plus a unique
      Port Number for this Client, then forwards the message into the

OMNI link spanning tree. (Note: including a unique Port Number
allows the Proxy/Server to distinguish different Clients located
behind the same Proxy at the link-layer, whereas the link-layer
addresses would otherwise be indistinguishable)

o when the Proxy/Server receives the RS, it authenticates the
   message then creates or updates a neighbor cache entry for the
   Client with the Proxy's ADM-ULA, INET address and Port Number as
   the link-layer address information. The Proxy/Server then sends
   an RA message back to the Proxy via the spanning tree.

o when the Proxy receives the RA, it authenticates the message and
   matches it with the proxy neighbor cache entry created by the RS.
   The Proxy then caches the prefix information as a mapping from the
   Client's MNPs to the Client's link-layer address, caches the
   Proxy/Server's advertised Router Lifetime and sets the neighbor
   cache entry state to REACHABLE. The Proxy then optionally
   rewrites the Router Lifetime and forwards the (proxyed) message to
   the Client. The Proxy finally includes an MTU option (if
   necessary) with an MTU to use for the underlying ANET interface.

After the initial RS/RA exchange, the Proxy forwards any of the
Client's carrier packets for which there is no matching neighbor
cache entry to a Bridge using OAL encapsulation with its own ADM-ULA
as the source and the MNP-ULA corresponding to the Client as the
destination. The Proxy instead forwards any carrier packets destined
to a neighbor cache target directly to the target according to the
OAL/link-layer information - the process of establishing neighbor
cache entries is specified in Section 3.14.

While the Client is still attached to the ANET, the Proxy sends NS,
RS and/or unsolicited NA messages to update the Proxy/Server's
neighbor cache entries on behalf of the Client and/or to convey QoS
updates. This allows for higher-frequency Proxy-initiated RS/RA
messaging over well-connected INET infrastructure supplemented by
lower-frequency Client-initiated RS/RA messaging over constrained
ANET data links.

If the Proxy/Server ceases to send solicited advertisements, the
Proxy sends unsolicited RAs on the ANET interface with destination
set to (link-local) All-Nodes multicast and with Router Lifetime set
to zero to inform Clients that the Proxy/Server has failed. Although
the Proxy engages in ND exchanges on behalf of the Client, the Client
can also send ND messages on its own behalf, e.g., if it is in a
better position than the Proxy to convey QoS changes, etc. For this
reason, the Proxy marks any Client-originated solicitation messages
(e.g. by inserting a Nonce option) so that it can return the

solicited advertisement to the Client instead of processing it
locally.

If the Client becomes unreachable, the Proxy sets the neighbor cache
entry state to DEPARTED and retains the entry for DepartTime seconds.
While the state is DEPARTED, the Proxy forwards any carrier packets
destined to the Client to a Bridge via OAL/ORH encapsulation. When
DepartTime expires, the Proxy deletes the neighbor cache entry and
discards any further carrier packets destined to this (now forgotten)
Client.

In some ANETs that employ a Proxy, the Client's MNP can be injected
into the ANET routing system. In that case, the Client can send
original IP packets without the OAL so that the ANET routing system
transports the original IP packets to the Proxy. This can be very
beneficial, e.g., if the Client connects to the ANET via low-end data
links such as some aviation wireless links.

If the first-hop ANET access router is on the same underlying link
and recognizes the AERO/OMNI protocol, the Client can avoid OAL
encapsulation for both its control and data messages. When the
Client connects to the link, it can send an unencapsulated RS message
with source address set to its own MNP-LLA (or to a Temporary LLA),
and with destination address set to the ADM-LLA of the Client's
selected Proxy/Server or to (link-local) All-Routers multicast. The
Client includes an OMNI option formatted as specified in
[I-D.templin-6man-omni-interface].

The Client then sends the unencapsulated RS message, which will be
intercepted by the AERO-Aware access router. The access router then
performs OAL encapsulation on the RS message with its own OAL/*NET
addresses as the source and the OAL/*NET addresses of a Proxy as the
destination. The access router further remembers the address of the
Proxy so that it can encapsulate future original IP packets from the
Client via the same Proxy. If the access router needs to change to a
new Proxy, it simply sends another RS message toward the Proxy/Server
via the new Proxy on behalf of the Client.

In some cases, the access router and Proxy may be one and the same
node. In that case, the node would be located on the same physical
link as the Client, but its message exchanges with the Proxy/Server
would need to pass through a security gateway at the ANET/INET
border. The method for deploying access routers and Proxys (i.e. as
a single node or multiple nodes) is an ANET-local administrative
consideration.

Note: The Proxy can apply packing as discussed in
[I-D.templin-6man-omni-interface] if an opportunity arises to

concatenate multiple original IP packets that will be destined to the
same neighbor.

3.13.1. Combined Proxy/Servers

Clients may need to connect directly to Proxy/Servers via INET,
Direct and VPNed interfaces (i.e., non-ANET interfaces). If the
Client's underlying interfaces all connect via the same INET
partition, then it can connect to a single controlling Proxy/Server
via all interfaces.

If some Client interfaces connect via different INET partitions,
however, the Client still selects a set of controlling Proxy/Servers
and sends RS messages via their directly-connected Proxy/Servers
while using the ADM-LLA of the controlling Proxy/Server as the
destination.

When a Proxy/Server receives an RS with destination set to the ADM-
LLA of a controlling Proxy/Server, it acts as a Proxy to forward the
message to the controlling Proxy/Server while forwarding the
corresponding RA reply to the Client.

3.13.2. Detecting and Responding to Proxy/Server Failures

In environments where fast recovery from Proxy/Server failure is
required, Proxys SHOULD use proactive Neighbor Unreachability
Detection (NUD) to track Server reachability in a similar fashion as
for Bidirectional Forwarding Detection (BFD) [RFC5880]. Proxys can
then quickly detect and react to failures so that cached information
is re-established through alternate paths. The NUD control messaging
is carried only over well-connected ground domain networks (i.e., and
not low-end aeronautical radio links) and can therefore be tuned for
rapid response.

Proxys perform proactive NUD with Proxy/Servers for which there are
currently active ANET Clients by sending continuous NS messages in
rapid succession, e.g., one message per second. The Proxy sends the
NS message via the spanning tree with the Proxy's ADM-LLA as the
source and the ADM-LLA of the Proxy/Server as the destination. When
the Proxy is also sending RS messages to the Proxy/Server on behalf
of ANET Clients, the resulting RA responses can be considered as
equivalent hints of forward progress. This means that the Proxy need
not also send a periodic NS if it has already sent an RS within the
same period. If the Proxy/Server fails (i.e., if the Proxy ceases to
receive advertisements), the Proxy can quickly inform Clients by
sending multicast RA messages on the ANET interface.

The Proxy sends RA messages on the ANET interface with source address
set to the Proxy/Server's address, destination address set to (link-
local) All-Nodes multicast, and Router Lifetime set to 0. The Proxy
SHOULD send MAX_FINAL_RTR_ADVERTISEMENTS RA messages separated by
small delays [RFC4861]. Any Clients on the ANET that had been using
the failed Proxy/Server will receive the RA messages and associate
with a new Proxy/Server.

3.13.3. Point-to-Multipoint Server Coordination

In environments where Client messaging over ANETs is bandwidth-
limited and/or expensive, Clients can enlist the services of the
Proxy to coordinate with multiple Proxy/Servers in a single RS/RA
message exchange. The Client can send a single RS message to (link-
local) All-Routers multicast that includes the ID's of multiple
Proxy/Servers in MS-Register sub-options of the OMNI option.

When the Proxy receives the RS and processes the OMNI option, it
sends a separate RS to each MS-Register Proxy/Server ID. When the
Proxy receives an RA, it can optionally return an immediate
"singleton" RA to the Client or record the Proxy/Server's ID for
inclusion in a pending "aggregate" RA message. The Proxy can then
return aggregate RA messages to the Client including multiple Proxy/
Server IDs in order to conserve bandwidth. Each RA includes a proper
subset of the Proxy/Server IDs from the original RS message, and the
Proxy must ensure that the message contents of each RA are consistent
with the information received from the (aggregated) Proxy/Servers.

Clients can thereafter employ efficient point-to-multipoint Proxy/
Server coordination under the assistance of the Proxy to reduce the
number of messages sent over the ANET while enlisting the support of
multiple Proxy/Servers for fault tolerance. Clients can further
include MS-Release sub-options in IPv6 ND messages to request the
Proxy to release from former Proxy/Servers via the procedures
discussed in Section 3.16.5.

The OMNI interface specification [I-D.templin-6man-omni-interface]
provides further discussion of the Client/Proxy RS/RA messaging
involved in point-to-multipoint coordination.

3.14. AERO Address Resolution

While carrier packets are flowing between a source and target node,
route optimization SHOULD be used. Route optimization is initiated
by the first eligible Route Optimization Source (ROS) closest to the
source as follows:

o For Clients on VPNed and Direct interfaces, the Server is the ROS.

   o For Clients on ANET interfaces, the Proxy is the ROS.

   o For Clients on INET interfaces, the Client itself is the ROS.

   o For correspondent nodes on INET/EUN interfaces serviced by a
      Relay, the Relay is the ROS.

   The route optimization procedure is conducted between the ROS and
   with the target Proxy/Server/Relay or the target Client itself acting
   as a Route Optimization Responder (ROR) in the same manner as for
   IPv6 ND Address Resolution and using the same NS/NA messaging. The
   target may either be a MNP Client serviced by a Proxy/Server, or a
   non-MNP correspondent reachable via a Relay.

   The procedures are specified in the following sections.

3.14.1. Route Optimization Initiation

   When an original IP packet from a source node destined to a target
   node arrives, the ROS checks for a neighbor cache entry for the LLA
   that matches the target destination. If there is a neighbor cache
   entry in the REACHABLE state, the ROS forwards the packets according
   to the cached state and returns from processing. Otherwise, if there
   is already a neighbor cache entry in the STALE state the ROS
   continues, and if there or no neighbor cache entry the ROS creates
   one in the INCOMPLETE state.

   The ROS next places the original IP packet on a short queue then
   sends an NS message for Address Resolution (NS(AR)) to receive a
   solicited NA(AR) message from a ROR. The NS(AR) message must be no
   larger than the OAL minimum MPS so that its entire contents will fit
   in an OAL first fragment (i.e., as an "atomic fragment"). The ROS
   prepares an NS(AR) that includes:

   o the LLA of the ROS as the source address.

   o the LLA corresponding to the original IP packet's destination as
      the Target Address, e.g., for 2001:db8:1:2::10:2000 the LLA is
      fe80::2001:db8:1:2.

   o the Solicited-Node multicast address [RFC4291] formed from the
      lower 24 bits of the original IP packet's destination as the
      destination address, e.g., for 2001:db8:1:2::10:2000 the NS
      destination address is ff02:0:0:0:0:1:ff10:2000.

   The NS(AR) message also includes an OMNI option with an Interface
   Attributes entry for the sending interface, and with Prefix Length
   set to the length associated with the ROS's LLA. The ROS then

submits the NS(AR) message for OAL encapsulation and fragmentation,
with OAL source set to its own ULA and OAL destination set to the ULA
corresponding to the target, and with an unpredictable initial
Identification value selected according to [RFC7739]. The source
caches the initial Identification value in the (newly-created)
neighbor cache entry as the starting sequence number for the "send"
window for future carrier packets sent to this target.

The ROS then sends the resulting carrier packet into the spanning
tree without decrementing the network-layer TTL/Hop Limit field.
(When the ROS is a Client, it instead must first sign the NS(AR)
message and send the resulting carrier packet to one of its current
Proxy/Servers which then verifies the NS(AR) signature and forwards
the carrier packet into the spanning tree on behalf of the Client.)

3.14.2. Relaying the NS(AR) *NET Packet(s)

When the Bridge receives the carrier packet from the ROS, it discards
the *NET headers and determines the next hop by consulting its
standard IPv6 forwarding table for the OAL header destination
address. The Bridge then re-encapsulates and forwards the carrier
packet via the spanning tree the same as for any IPv6 router, where
it may traverse multiple OMNI link segments. The final-hop Bridge in
the spanning tree will deliver the carrier packet via a secured
tunnel to a Proxy/Server or Relay that services the target.

3.14.3. Processing the NS(AR) and Sending the NA(AR)

When the Proxy/Server/Relay receives the carrier packet, it examines
the enclosed atomic OAL fragment to determine that it contains an
NS(AR) then examines the NS(AR) target to determine whether it has a
matching neighbor cache entry and/or non-MNP route. If there is no
match, the Proxy/Server/Relay drops the message. Otherwise, the ROR
continues processing as follows:

o if the NS(AR) target matches a Client neighbor cache entry in the
   DEPARTED state, the Proxy/Server inserts an ORH with destination
   prefix set to the lower 64 bits of the Client's MNP-ULA and sets
   the destination address to the ADM-ULA of the Client's new Proxy/
   Server. The (old) Proxy/Server then re-encapsulates the carrier
   packet, forwards it into the spanning tree and returns from
   processing.

o If the NS(AR) target matches a Client neighbor cache entry in the
   REACHABLE state, the Proxy/Server proceeds according to whether
   the Client requires Proxy services. For "disadvantaged" Clients
   (e.g., those on low-end links) and Clients with complex multilink
   arrangements the Proxy/Server nominates itself as the ROR;

otherwise, the Proxy/Server re-encapsulates the carrier packet and
forwards it to the target Client which will act as an ROR on its
own behalf.

o If the NS(AR) target matches one of its non-MNP routes, the Relay
   acts the ROR.

The ROR next checks for a neighbor cache entry that matches the
NS(AR) source LLA. If there is a neighbor cache entry in the
REACHABLE state, the ROR accepts the NS(AR) only if the OAL
Identification value is within the "accept" window for this NS(AR)
source LLA or if the NS(AR) was forwarded by the (secured) spanning
tree. Otherwise, the ROR creates a neighbor cache entry for the
NS(AR) source LLA if necessary with state set to STALE. If the ROR
is a Proxy/Server, it next adds a Report List entry to the target
Client neighbor cache entry for the NS(AR) source LLA while caching
the OAL Identification value in the (newly-created) neighbor cache
entry as the starting sequence number for the "accept" window for
future carrier packets received from this LLA source.

The ROR then prepares a (solicited) NA(AR) message to send back to
the ROS using the same Identification value received in the NS(AR)
(unlike the NS(AR), the NA(AR) need not fit in a single OAL
fragment). The ROR sets the NA(AR) source address to its own LLA,
sets the destination address to the NS(AR) LLA source address and
sets the Target Address to the same value that was in the NS(AR).
The ROR then includes an OMNI option with Prefix Length set to the
length associated with the MNP-LLA.

If the NS(AR) target was an MNP Client, the ROR next includes
Interface Attributes in the OMNI option for each of the target
Client's underlying interfaces with current information for each
interface. If the ROR is a Proxy/Server/Relay, it then sets the S/
T-ifIndex field in the OMNI header set to 0. If the ROR is the
Client itself, it instead sets S/T-ifIndex to the index of the
underlying interface that will send the NS(AR).

For each Interface Attributes sub-option, the ROR sets the L2ADDR
according to the Proxy/Server's INET address for VPNed or Direct
interfaces, to the INET address of the Proxy for Proxyed interfaces
or to the Client's INET address for INET interfaces. The ROR then
includes the lower 32 bits of the Proxy/Server's ADM-ULA as the LHS,
encodes the ADM-ULA prefix length code in the SRT field and sets the
FMT code accordingly as specified in Section 3.3.

The ROR then sets the NA(AR) message R flag to 1 (as a router) and S
flag to 1 (as a response to a solicitation). If the ROR is the
Client itself, it sets the O flag to 1; if the ROR is the Proxy/

Server, it instead sets the O flag to 0 (as a proxy). The ROR
finally submits the NA(AR) for OAL encapsulation with source set to
its own ULA and destination set to the source ULA of the NS(AR)
message, then performs OAL fragmentation using the same
Identification value that appeared in the NS(AR) and forwards the
resulting (*NET-encapsulated) carrier packets into the spanning tree
without decrementing the network-layer TTL/Hop Limit field.

3.14.4. Relaying the NA(AR)

When the Bridge receives the carrier packets from the ROR, it
discards the *NET header and determines the next hop by consulting
its standard IPv6 forwarding table for the OAL header destination
address. The Bridge then re-encapsulates and forwards the carrier
packets via the spanning tree the same as for any IPv6 router, where
it may traverse multiple OMNI link segments. The final-hop Bridge in
the spanning tree will deliver the carrier packet via a secured
tunnel to a Proxy/Server for the ROS.

3.14.5. Processing the NA(AR)

When the ROS receives the NA(AR) message, it first searches for a
neighbor cache entry that matches the NA(AR) target address. If
there is an entry in the INCOMPLETE or STALE state, the ROS matches
the OAL Identification value with the value it had included in the
corresponding NS(AR). If the values match, the ROS processes the
message the same as for standard IPv6 Address Resolution [RFC4861].
In the process, it caches the target LLA and all information found in
the OMNI option in the neighbor cache entry for the target. The ROS
finally sets the neighbor cache entry state to REACHABLE and sets its
lifetime to ReachableTime seconds. (When the ROS is a Client, the
solicited NA(AR) message will first be delivered via the spanning
tree to one of its current Proxy/Servers, which then securely
forwards the message to the Client.)

3.14.6. Route Optimization Maintenance

Following route optimization, the ROS forwards future carrier packets
with user data destined to the target via the addresses found in the
cached link-layer information. The route optimization is shared by
all sources that send original IP packets to the target via the ROS,
i.e., and not just the source on behalf of which the route
optimization was initiated. Note that route optimization is
performed only for original IP packets that contain user data, and
not for those that contain other IPv6 ND control messages.

While the ROS continues to forward additional original IP packets
destined to the target, it sends additional NS(AR) messages to the

ROR before ReachableTime expires to receive a fresh NA(AR) message
the same as described in the previous sections (route optimization
refreshment strategies are an implementation matter, with a non-
normative example given in Appendix A.1). The ROS may supply a new
unpredictable OAL Identification value if it wishes to reset the
neighbor's "accept" Identification window. If the ROS is a Client,
it must sign the NS(AR) message so that the Proxy/Server can
authenticate.

The ROS uses the cached ULA of the ROR (i.e., either the ADM-ULA of
the Proxy/Server or the MNP-ULA of the Client itself) as the NS(AR)
OAL destination address, and sends up to MAX_MULTICAST_SOLICIT NS(AR)
messages separated by 1 second until an NA(AR) is received. If no
NA(AR) is received, the ROS assumes that the current ROR has become
unreachable and deletes the target neighbor cache entry. Subsequent
original IP packets will trigger a new route optimization event (see:
Section 3.14.1).

If an NA(AR) is received, the ROS then updates the neighbor cache
entry to refresh ReachableTime, while (for MNP destinations) the ROR
adds or updates the ROS address to the target's Report List and with
time set to ReportTime. While no carrier packets are flowing, the
ROS instead allows ReachableTime for the neighbor cache entry to
expire. When ReachableTime expires, the ROS deletes the neighbor
cache entry. Any future carrier packets flowing through the ROS will
again trigger a new route optimization.

The ROS may also receive unsolicited NA messages from the ROR at any
time (see: Section 3.16). If there is a neighbor cache entry for the
target, the ROS updates the link-layer information but does not
update ReachableTime since the receipt of an unsolicited NA does not
confirm that any forward paths are working. If there is no neighbor
cache entry, the ROS simply discards the unsolicited NA.

In this arrangement, the ROS holds a neighbor cache entry with only a
"send" Identification window value for the target, while the ROR
holds a neighbor cache entry with only an "accept" Identification
window value for the ROS. The route optimization neighbor
relationship is therefore asymmetric and unidirectional. If the
target node also has carrier packets to send back to the source node,
then a separate route optimization procedure is performed in the
reverse direction. But, there is no requirement that the forward and
reverse paths be symmetric.

3.15. **Neighbor Unreachability Detection (NUD)**

   AERO nodes perform Neighbor Unreachability Detection (NUD) per
   [RFC4861] either reactively in response to persistent link-layer
   errors (see Section 3.11) or proactively to confirm reachability and/
   or establish NAT state. The NUD algorithm is based on periodic
   control message exchanges. The algorithm may further be seeded by ND
   hints of forward progress, but care must be taken to avoid inferring
   reachability based on spoofed information. For example, authentic
   IPv6 ND message exchanges may be considered as acceptable hints of
   forward progress, while spurious random packets should not be.

   AERO nodes can use standard NS/NA exchanges sent over the OMNI link
   spanning tree to securely test reachability without risk of DoS
   attacks from nodes pretending to be a neighbor (these NS/NA(NUD)
   messages use the unicast LLAs and ULAs of the two parties involved in
   the NUD test the same as for standard IPv6 ND, and both messages flow
   over the spanning tree). Proxys can further perform NUD to securely
   verify Proxy/Server reachability on behalf of their proxyed Clients.
   However, a means for an ROS to test the unsecured target route
   optimized paths is also necessary.

   When an ROR directs an ROS to a target neighbor with one or more
   link-layer addresses, the ROS can proactively test each such
   unsecured route optimized path through secured NS(NUD) messages over
   the spanning tree that invoke an unsecured NA(NUD) reply that travels
   over the route optimized path. (The NS(NUD) messages must therefore
   include Nonce and Timestamp options that will be echoed in the
   unsecured NA(NUD) replies.) While testing the paths, the ROS can
   optionally continue to send carrier packets via the spanning tree,
   maintain a small queue of carrier packets until target reachability
   is confirmed, or (optimistically) allow carrier packets to flow via
   the route optimized paths.

   When the ROS sends an NS(NUD) message, it sets the IPv6 source to its
   own address, sets the destination to the MNP-LLA of the target, and
   sets the target's MNP Subnet-Router anycast address as the Target
   Address. The ROS also includes an OMNI option with a single
   Interface Attributes sub-option with the SRT, FMT, LHS and L2ADDR
   information for its own underlying interface it wishes to test, but
   sets the S/T-ifIndex field to the index for target's underlying
   interface to be tested. The ROS includes a Nonce and Timestamp
   option, then encapsulates the message in OAL/INET headers with its
   own ULA as the source and the ULA of the target as the destination.
   The ROS then forwards the NS(NUD) message toward the target via a
   Proxy/Server or Bridge.

When the target receives the NS(NUD) message, it creates an NA(NUD)
by reversing the OAL and IPv6 addresses and including an Interface
Attributes sub-option with attributes for its own interface
identified by the NS(NUD) S/T-ifIndex. The target sets the NA(NUD)
S/T-ifIndex to the index of the ROS, sets the Target Address to the
same value that was in the NS(NUD), and returns the message using its
own underlying interface identified by S/T-ifIndex and destined to
the ROS's interface identified by the original Interface Attributes
sub-option.

When the ROS receives the NA(NUD) message, it can determine from the
Nonce, Timestamp and Target Address that the message matched its
NS(NUD) and that it transited the direct path from the ROR using the
selected underlying interface pair. The ROS marks route optimization
target paths that pass these NUD tests as "reachable", and those that
do not as "unreachable". These markings inform the OMNI interface
forwarding algorithm specified in Section 3.10.

Note: If the target determines that the OMNI option Interface
Attributes in the NS(NUD) is located in a different OMNI link segment
than its own interface named in the S/T-ifIndex, it instead returns
the NA(NUD) via the spanning tree while including an ORH and setting
the OAL destination address to the Subnet Router Anycast address used
by Bridges on the ROS segment. When a Bridge on the ROS segment
receives the NA(NUD), it replaces the Interface Attributes with
information for its own interface while using the ifIndex value
specific to the target.

3.16. Mobility Management and Quality of Service (QoS)

AERO is a Distributed Mobility Management (DMM) service. Each Proxy/
Server is responsible for only a subset of the Clients on the OMNI
link, as opposed to a Centralized Mobility Management (CMM) service
where there is a single network mobility collective entity for all
Clients. Clients coordinate with their associated Proxy/Servers via
RS/RA exchanges to maintain the DMM profile, and the AERO routing
system tracks all current Client/Proxy/Server peering relationships.

Proxy/Servers provide default routing and mobility/multilink services
for their dependent Clients. Clients are responsible for maintaining
neighbor relationships with their Proxy/Servers through periodic RS/
RA exchanges, which also serves to confirm neighbor reachability.
When a Client's underlying interface address and/or QoS information
changes, the Client is responsible for updating the Proxy/Server with
this new information. Note that when there is a Proxy in the path,
the Proxy can also perform some RS/RA exchanges on the Client's
behalf.

Mobility management messaging is based on the transmission and
reception of unsolicited Neighbor Advertisement (uNA) messages. Each
uNA message sets the IPv6 destination address to (link-local) All-
Nodes multicast to convey a general update of Interface Attributes to
(possibly) multiple recipients, or to a specific unicast LLA to
announce a departure event to a specific recipient. Implementations
must therefore examine the destination address to determine the
nature of the mobility event (i.e., update vs departure).

Mobility management considerations are specified in the following
sections.

3.16.1. Mobility Update Messaging

Proxy/Servers accommodate Client mobility, multilink and/or QoS
change events by sending unsolicited NA (uNA) messages to each ROS in
the target Client's Report List. When a Proxy/Server sends a uNA
message, it sets the IPv6 source address to the Client's MNP-LLA,
sets the destination address to (link-local) All-Nodes multicast and
sets the Target Address to the Client's Subnet-Router anycast
address. The Proxy/Server also includes an OMNI option with Prefix
Length set to the length associated with the Client's MNP-LLA, with
Interface Attributes for the target Client's underlying interfaces
and with the OMNI header S/T-ifIndex set to 0. The Proxy/Server then
sets the NA R flag to 1, the S flag to 0 and the O flag to 1, then
encapsulates the message in an OAL header with source set to its own
ADM-ULA and destination set to the ULA of the ROS and sends the
message into the spanning tree.

As discussed in Section 7.2.6 of [RFC4861], the transmission and
reception of uNA messages is unreliable but provides a useful
optimization. In well-connected Internetworks with robust data links
uNA messages will be delivered with high probability, but in any case
the Proxy/Server can optionally send up to MAX_NEIGHBOR_ADVERTISEMENT
uNAs to each ROS to increase the likelihood that at least one will be
received.

When the ROS receives a uNA message prepared as above, it ignores the
message if there is no existing neighbor cache entry for the Client.
Otherwise, it uses the included OMNI option information to update the
neighbor cache entry, but does not reset ReachableTime since the
receipt of an unsolicited NA message from the target Proxy/Server
does not provide confirmation that any forward paths to the target
Client are working.

If uNA messages are lost, the ROS may be left with stale address and/
or QoS information for the Client for up to ReachableTime seconds.
During this time, the ROS can continue sending carrier packets

according to its stale neighbor cache information. When
ReachableTime is close to expiring, the ROS will re-initiate route
optimization and receive fresh link-layer address information.

In addition to sending uNA messages to the current set of ROSs for
the Client, the Proxy/Server also sends uNAs to the MNP-ULA
associated with the link-layer address for any underlying interface
for which the link-layer address has changed. These uNA messages
update an old Proxy/Server that cannot easily detect (e.g., without
active probing) when a formerly-active Client has departed. When the
Proxy/Server sends the uNA, it sets the IPv6 source address to the
Client's MNP-LLA, sets the destination address to the old Proxy/
Server's ADM-LLA, and sets the Target Address to the Client's Subnet-
Router anycast address. The Proxy/Server also includes an OMNI
option with Prefix Length set to the length associated with the
Client's MNP-LLA, with Interface Attributes for the changed
underlying interface, and with the OMNI header S/T-ifIndex set to 0.
The Proxy/Server then sets the NA R flag to 1, the S flag to 0 and
the O flag to 1, then encapsulates the message in an OAL header with
source set to its own ADM-ULA and destination set to the ADM-ULA of
the old Proxy/Server and sends the message into the spanning tree.

[3.16.2]. Announcing Link-Layer Address and/or QoS Preference Changes

When a Client needs to change its underlying interface addresses and/
or QoS preferences (e.g., due to a mobility event), either the Client
or its Proxys send RS messages to the Proxy/Server via the spanning
tree with an OMNI option that includes Interface attributes with the
new link quality and address information.

Up to MAX_RTR_SOLICITATIONS RS messages MAY be sent in parallel with
sending carrier packets containing user data in case one or more RAs
are lost. If all RAs are lost, the Client SHOULD re-associate with a
new Proxy/Server.

When the Proxy/Server receives the Client's changes, it sends uNA
messages to all nodes in the Report List the same as described in the
previous section.

[3.16.3]. Bringing New Links Into Service

When a Client needs to bring new underlying interfaces into service
(e.g., when it activates a new data link), it sends an RS message to
the Proxy/Server via the underlying interface with an OMNI option
that includes Interface Attributes with appropriate link quality
values and with link-layer address information for the new link.

3.16.4. **Deactivating Existing Links**

   When a Client needs to deactivate an existing underlying interface,
   it sends an RS or uNA message to its Proxy/Server with an OMNI option
   with appropriate Interface Attribute values - in particular, the link
   quality value 0 assures that neighbors will cease to use the link.

   If the Client needs to send RS/uNA messages over an underlying
   interface other than the one being deactivated, it MUST include
   Interface Attributes with appropriate link quality values for any
   underlying interfaces being deactivated.

   Note that when a Client deactivates an underlying interface,
   neighbors that have received the RS/uNA messages need not purge all
   references for the underlying interface from their neighbor cache
   entries. The Client may reactivate or reuse the underlying interface
   and/or its ifIndex at a later point in time, when it will send RS/uNA
   messages with fresh Interface Attributes to update any neighbors.

3.16.5. **Moving Between Servers**

   The Client performs the procedures specified in Section 3.12.2 when
   it first associates with a new Proxy/Server or renews its association
   with an existing Proxy/Server. The Client also includes MS-Release
   identifiers in the RS message OMNI option per
   [I-D.templin-6man-omni-interface] if it wants the new Proxy/Server to
   notify any old Proxy/Servers from which the Client is departing.

   When the new Proxy/Server receives the Client's RS message, it
   returns an RA as specified in Section 3.12.3 and sends up to
   MAX_NEIGHBOR_ADVERTIISEMENT uNA messages to any old Proxy/Servers
   listed in OMNI option MS-Release identifiers. When the new Proxy/
   Server sends a uNA message, it sets the IPv6 source address to the
   Client's MNP-LLA, sets the destination address to the old Proxy/
   Server's ADM-LLA, and sets the Target Address to the Client's Subnet-
   Router anycast address. The new Proxy/Server also includes an OMNI
   option with Prefix Length set to the length associated with the
   Client's MNP-LLA, with Interface Attributes for its own underlying
   interface, and with the OMNI header S/T-ifIndex set to 0. The new
   Proxy/Server then sets the NA R flag to 1, the S flag to 0 and the O
   flag to 1, then encapsulates the message in an OAL header with source
   set to its own ADM-ULA and destination set to the ADM-ULA of the old
   Proxy/Server and sends the message into the spanning tree.

   When an old Proxy/Server receives the uNA, it changes the Client's
   neighbor cache entry state to DEPARTED, sets the link-layer address
   of the Client to the new Proxy/Server's ADM-ULA, and resets
   DepartTime. After a short delay (e.g., 2 seconds) the old Proxy/

Server withdraws the Client's MNP from the routing system. After
DepartTime expires, the old Proxy/Server deletes the Client's
neighbor cache entry.

The old Proxy/Server also iteratively forwards a copy of the uNA
message to each ROS in the Client's Report List by changing the OAL
destination address to the ULA of the ROS while leaving all other
fields of the message unmodified. When the ROS receives the uNA, it
examines the Target address to determine the correct neighbor cache
entry and verifies that the IPv6 destination address matches the old
Proxy/Server. The ROS then caches the IPv6 source address as the new
Proxy/Server for the existing neighbor cache entry and marks the
entry as STALE. While in the STALE state, the ROS allows new carrier
packets to flow according to any existing cached link-layer
information and sends new NS(AR) messages using its own ULA as the
OAL source and the ADM-ULA of the new Proxy/Server as the OAL
destination address to elicit NA messages that reset the neighbor
cache entry state to REACHABLE. If no new NA message is received for
10 seconds while in the STALE state, the ROS deletes the neighbor
cache entry.

Clients SHOULD NOT move rapidly between Proxy/Servers in order to
avoid causing excessive oscillations in the AERO routing system.
Examples of when a Client might wish to change to a different Proxy/
Server include a Proxy/Server that has gone unreachable, topological
movements of significant distance, movement to a new geographic
region, movement to a new OMNI link segment, etc.

When a Client moves to a new Proxy/Server, some of the fragments of a
multiple fragment OAL packet may have already arrived at the old
Proxy/Server while others are en route to the new Proxy/Server,
however no special attention in the reassembly algorithm is necessary
since all fragments will eventually be delivered to the Client which
can then reassemble.

3.17. Multicast

The AERO Client provides an IGMP (IPv4) [RFC2236] or MLD (IPv6)
[RFC3810] proxy service for its EUNs and/or hosted applications
[RFC4605]. The Client forwards IGMP/MLD messages over any of its
underlying interfaces for which group membership is required. The
IGMP/MLD messages may be further forwarded by a first-hop ANET access
router acting as an IGMP/MLD-snooping switch [RFC4541], then
ultimately delivered to an AERO Proxy/Server acting as a Protocol
Independent Multicast - Sparse-Mode (PIM-SM, or simply "PIM")
Designated Router (DR) [RFC7761]. AERO Relays also act as PIM
routers (i.e., the same as AERO Proxys/Servers) on behalf of nodes on
INET/EUN networks. The behaviors identified in the following

sections correspond to Source-Specific Multicast (SSM) and Any-Source
Multicast (ASM) operational modes.

3.17.1. Source-Specific Multicast (SSM)

When an ROS (i.e., an AERO Proxy/Server/Relay) "X" acting as PIM
router receives a Join/Prune message from a node on its downstream
interfaces containing one or more ((S)ource, (G)roup) pairs, it
updates its Multicast Routing Information Base (MRIB) accordingly.
For each S belonging to a prefix reachable via X's non-OMNI
interfaces, X then forwards the (S, G) Join/Prune to any PIM routers
on those interfaces per [RFC7761].

For each S belonging to a prefix reachable via X's OMNI interface, X
originates a separate copy of the Join/Prune for each (S,G) in the
message using its own LLA as the source address and ALL-PIM-ROUTERS
as the destination address. X then encapsulates each message in an
OAL header with source address set to the ULA of X and destination
address set to S then forwards the message into the spanning tree,
which delivers it to AERO Proxy/Server/Relay "Y" that services S. At
the same time, if the message was a Join, X sends a route-
optimization NS message toward each S the same as discussed in
Section 3.14. The resulting NAs will return the LLA for the prefix
that matches S as the network-layer source address and with an OMNI
option with the ULA corresponding to any underlying interfaces that
are currently servicing S.

When Y processes the Join/Prune message, if S located behind any
INET, Direct, or VPNed interfaces Y acts as a PIM router and updates
its MRIB to list X as the next hop in the reverse path. If S is
located behind any Proxys "Z"*, Y also forwards the message to each
Z* over the spanning tree while continuing to use the LLA of X as the
source address. Each Z* then updates its MRIB accordingly and
maintains the LLA of X as the next hop in the reverse path. Since
the Bridges do not examine network layer control messages, this means
that the (reverse) multicast tree path is simply from each Z* (and/or
Y) to X with no other multicast-aware routers in the path. If any Z*
(and/or Y) is located on the same OMNI link segment as X, the
multicast data traffic sent to X directly using OAL/INET
encapsulation instead of via a Bridge.

Following the initial Join/Prune and NS/NA messaging, X maintains a
neighbor cache entry for each S the same as if X was sending unicast
data traffic to S. In particular, X performs additional NS/NA
exchanges to keep the neighbor cache entry alive for up to t_periodic
seconds [RFC7761]. If no new Joins are received within t_periodic
seconds, X allows the neighbor cache entry to expire. Finally, if X
receives any additional Join/Prune messages for (S,G) it forwards the

   messages to each Y and Z* in the neighbor cache entry over the
   spanning tree.

   At some later time, Client C that holds an MNP for source S may
   depart from a first Proxy Z1 and/or connect via a new Proxy Z2. In
   that case, Y sends an unsolicited NA message to X the same as
   specified for unicast mobility in Section 3.16. When X receives the
   unsolicited NA message, it updates its neighbor cache entry for the
   LLA for source S and sends new Join messages to any new Proxys Z2.
   There is no requirement to send any Prune messages to old Proxys Z1
   since source S will no longer source any multicast data traffic via
   Z1. Instead, the multicast state for (S,G) in Proxy Z1 will soon
   time out since no new Joins will arrive.

   After some later time, C may move to a new Proxy/Server Y2 and depart
   from old Sever Y1. In that case, Y1 sends Join messages for any of
   C's active (S,G) groups to Y2 while including its own LLA as the
   source address. This causes Y2 to include Y1 in the multicast
   forwarding tree during the interim time that Y1's neighbor cache
   entry for C is in the DEPARTED state. At the same time, Y1 sends an
   unsolicited NA message to X with an OMNI option with S/T-ifIndex in
   the header set to 0 and a release indication to cause X to release
   its neighbor cache entry. X then sends a new Join message to S via
   the spanning tree and re-initiates route optimization the same as if
   it were receiving a fresh Join message from a node on a downstream
   link.

3.17.2. Any-Source Multicast (ASM)

   When an ROS X acting as a PIM router receives a Join/Prune from a
   node on its downstream interfaces containing one or more (*,G) pairs,
   it updates its Multicast Routing Information Base (MRIB) accordingly.
   X then forwards a copy of the message to the Rendezvous Point (RP) R
   for each G over the spanning tree. X uses its own LLA as the source
   address and ALL-PIM-ROUTERS as the destination address, then
   encapsulates each message in an OAL header with source address set to
   the ULA of X and destination address set to R, then sends the message
   into the spanning tree. At the same time, if the message was a Join
   X initiates NS/NA route optimization the same as for the SSM case
   discussed in Section 3.17.1.

   For each source S that sends multicast traffic to group G via R, the
   Proxy/Server Z* for the Client that aggregates S encapsulates the
   original IP packets in PIM Register messages and forwards them to R
   via the spanning tree, which may then elect to send a PIM Join to Z*.
   This will result in an (S,G) tree rooted at Z* with R as the next hop
   so that R will begin to receive two copies of the original IP packet;
   one native copy from the (S, G) tree and a second copy from the pre-

existing (*, G) tree that still uses PIM Register encapsulation. R
can then issue a PIM Register-stop message to suppress the Register-
encapsulated stream. At some later time, if C moves to a new Proxy/
Server Z*, it resumes sending original IP packets via PIM Register
encapsulation via the new Z*.

At the same time, as multicast listeners discover individual S's for
a given G, they can initiate an (S,G) Join for each S under the same
procedures discussed in Section 3.17.1. Once the (S,G) tree is
established, the listeners can send (S, G) Prune messages to R so
that multicast original IP packets for group G sourced by S will only
be delivered via the (S, G) tree and not from the (*, G) tree rooted
at R. All mobility considerations discussed for SSM apply.

3.17.3. Bi-Directional PIM (BIDIR-PIM)

Bi-Directional PIM (BIDIR-PIM) [RFC5015] provides an alternate
approach to ASM that treats the Rendezvous Point (RP) as a Designated
Forwarder (DF). Further considerations for BIDIR-PIM are out of
scope.

3.18. Operation over Multiple OMNI Links

An AERO Client can connect to multiple OMNI links the same as for any
data link service. In that case, the Client maintains a distinct
OMNI interface for each link, e.g., 'omni0' for the first link,
'omni1' for the second, 'omni2' for the third, etc. Each OMNI link
would include its own distinct set of Bridges and Proxy/Servers,
thereby providing redundancy in case of failures.

Each OMNI link could utilize the same or different ANET connections.
The links can be distinguished at the link-layer via the SRT prefix
in a similar fashion as for Virtual Local Area Network (VLAN) tagging
(e.g., IEEE 802.1Q) and/or through assignment of distinct sets of
MSPs on each link. This gives rise to the opportunity for supporting
multiple redundant networked paths, with each VLAN distinguished by a
different SRT "color" (see: Section 3.2.5).

The Client's IP layer can select the outgoing OMNI interface
appropriate for a given traffic profile while (in the reverse
direction) correspondent nodes must have some way of steering their
original IP packets destined to a target via the correct OMNI link.

In a first alternative, if each OMNI link services different MSPs,
then the Client can receive a distinct MNP from each of the links.
IP routing will therefore assure that the correct OMNI link is used
for both outbound and inbound traffic. This can be accomplished

using existing technologies and approaches, and without requiring any
special supporting code in correspondent nodes or Bridges.

In a second alternative, if each OMNI link services the same MSP(s)
then each link could assign a distinct "OMNI link Anycast" address
that is configured by all Bridges on the link. Correspondent nodes
can then perform Segment Routing to select the correct SRT, which
will then direct the original IP packet over multiple hops to the
target.

3.19. DNS Considerations

AERO Client MNs and INET correspondent nodes consult the Domain Name
System (DNS) the same as for any Internetworking node. When
correspondent nodes and Client MNs use different IP protocol versions
(e.g., IPv4 correspondents and IPv6 MNs), the INET DNS must maintain
A records for IPv4 address mappings to MNs which must then be
populated in Relay NAT64 mapping caches. In that way, an IPv4
correspondent node can send original IPv4 packets to the IPv4 address
mapping of the target MN, and the Relay will translate the IPv4
header and destination address into an IPv6 header and IPv6
destination address of the MN.

When an AERO Client registers with an AERO Proxy/Server, the Proxy/
Server can return the address(es) of DNS servers in RDNSS options
[RFC6106]. The DNS server provides the IP addresses of other MNs and
correspondent nodes in AAAA records for IPv6 or A records for IPv4.

3.20. Transition Considerations

OAL encapsulation ensures that dissimilar INET partitions can be
joined into a single unified OMNI link, even though the partitions
themselves may have differing protocol versions and/or incompatible
addressing plans. However, a commonality can be achieved by
incrementally distributing globally routable (i.e., native) IP
prefixes to eventually reach all nodes (both mobile and fixed) in all
OMNI link segments. This can be accomplished by incrementally
deploying AERO Relays on each INET partition, with each Relay
distributing its MNPs and/or discovering non-MNP IP GUA prefixes on
its INET links.

This gives rise to the opportunity to eventually distribute native IP
addresses to all nodes, and to present a unified OMNI link view even
if the INET partitions remain in their current protocol and
addressing plans. In that way, the OMNI link can serve the dual
purpose of providing a mobility/multilink service and a transition
service. Or, if an INET partition is transitioned to a native IP
protocol version and addressing scheme that is compatible with the

OMNI link MNP-based addressing scheme, the partition and OMNI link
can be joined by Relays.

Relays that connect INETs/EUNs with dissimilar IP protocol versions
may need to employ a network address and protocol translation
function such as NAT64 [RFC6146].

3.21. Detecting and Reacting to Server and Bridge Failures

In environments where rapid failure recovery is required, Proxy/
Servers and Bridges SHOULD use Bidirectional Forwarding Detection
(BFD) [RFC5880]. Nodes that use BFD can quickly detect and react to
failures so that cached information is re-established through
alternate nodes. BFD control messaging is carried only over well-
connected ground domain networks (i.e., and not low-end radio links)
and can therefore be tuned for rapid response.

Proxy/Servers and Bridges maintain BFD sessions in parallel with
their BGP peerings. If a Proxy/Server or Bridge fails, BGP peers
will quickly re-establish routes through alternate paths the same as
for common BGP deployments. Similarly, Proxys maintain BFD sessions
with their associated Bridges even though they do not establish BGP
peerings with them.

Proxys SHOULD use proactive NUD for Proxy/Servers for which there are
currently active ANET Clients in a manner that parallels BFD, i.e.,
by sending unicast NS messages in rapid succession to receive
solicited NA messages. When the Proxy is also sending RS messages on
behalf of ANET Clients, the RS/RA messaging can be considered as
equivalent hints of forward progress. This means that the Proxy need
not also send a periodic NS if it has already sent an RS within the
same period. If a Proxy/Server fails, the Proxy will cease to
receive advertisements and can quickly inform Clients of the outage
by sending multicast RA messages on the ANET interface.

The Proxy sends multicast RA messages with source address set to the
Proxy/Server's address, destination address set to (link-local) All-
Nodes multicast, and Router Lifetime set to 0. The Proxy SHOULD send
MAX_FINAL_RTR_ADVERTISEMENTS RA messages separated by small delays
[RFC4861]. Any Clients on the ANET interface that have been using
the (now defunct) Proxy/Server will receive the RA messages and
associate with a new Proxy/Server.

3.22. AERO Clients on the Open Internet

AERO Clients that connect to the open Internet via INET interfaces
can establish a VPN or direct link to securely connect to a Proxy/
Server in a "tethered" arrangement with all of the Client's traffic

transiting the Proxy/Server. Alternatively, the Client can associate
with an INET Proxy/Server using UDP/IP encapsulation and control
message securing services as discussed in the following sections.

When a Client's OMNI interface enables an INET underlying interface,
it first determines whether the interface is likely to be behind a
NAT. For IPv4, the Client assumes it is on the open Internet if the
INET address is not a special-use IPv4 address per [RFC3330].
Similarly for IPv6, the Client assumes it is on the open Internet if
the INET address is a Global Unicast Address (GUA) [RFC4291].
Otherwise, the Client assumes it may be behind one or several NATs.

The Client then prepares an RS message with IPv6 source address set
to its MNP-LLA, with IPv6 destination set to (link-local) All-Routers
multicast and with an OMNI option with underlying interface
attributes. If the Client believes that it is on the open Internet,
it SHOULD include an L2ADDR in the Interface Attributes sub-option
corresponding to the underlying interface; otherwise, it MAY omit
L2ADDR. If the underlying address is IPv4, the Client includes the
Port Number and IPv4 address written in obfuscated form [RFC4380] as
discussed in Section 3.3. If the underlying interface address is
IPv6, the Client instead includes the Port Number and IPv6 address in
obfuscated form. The Client finally includes a HIP "Initiator"
message sub-option in the OMNI option
[I-D.templin-6man-omni-interface] to provide message authentication
and performs OAL encapsulation and fragmentation if necessary. The
Client then encapsulates each fragment in UDP/IP headers, sets the
UDP/IP source to its INET address and UDP port, sets the UDP/IP
destination to the Proxy/Server's INET address and the AERO service
port number (8060), then sends the (whole or fragmented) message to
the Proxy/Server.

When the Proxy/Server receives the RS, it performs OAL reassembly if
necessary then authenticates the message and registers the Client's
MNP and INET interface information according to the OMNI option
parameters. If the RS message OMNI option includes Interface
Attributes with an L2ADDR, the Proxy/Server compares the
encapsulation IP address and UDP port number with the (unobfuscated)
values. If the values are the same, the Proxy/Server caches the
Client's information as "INET" addresses meaning that the Client is
likely to accept direct messages without requiring NAT traversal
exchanges. If the values are different (or, if the OMNI option did
not include an L2ADDR) the Proxy/Server instead caches the Client's
information as "mapped" addresses meaning that NAT traversal
exchanges may be necessary.

The Proxy/Server then prepares an RA message with IPv6 source and
destination set corresponding to the addresses in the RS, and with an

OMNI option with an Origin Indication sub-option per
[I-D.templin-6man-omni-interface] with the mapped and obfuscated Port
Number and IP address observed in the encapsulation headers. The
Proxy/Server also includes a HIP "Responder" message sub-option per
[I-D.templin-6man-omni-interface] that contains an acknowledgement of
the update sent by the Client. The Proxy/Server then performs OAL
encapsulation and fragmentation if necessary, and encapsulates each
fragment in UDP/IP headers with addresses set per the L2ADDR
information in the neighbor cache entry for the Client.

When the Client receives the RA message, it performs OAL reassembly
if necessary, authenticates the HIP "Responder" message, then
compares the mapped Port Number and IP address from the Origin
Indication sub-option with its own address. If the addresses are the
same, the Client assumes the open Internet / Cone NAT principle; if
the addresses are different, the Client instead assumes that further
qualification procedures are necessary to detect the type of NAT and
proceeds according to standard procedures [RFC6081][RFC4380].

After the Client has registered its INET interfaces in such RS/RA
exchanges it sends periodic RS messages to receive fresh RA messages
before the Router Lifetime received on each INET interface expires.
The Client also maintains default routes via its Proxy/Servers, i.e.,
the same as described in earlier sections.

When the Client sends messages to target IP addresses, it also
invokes route optimization per Section 3.14 using IPv6 ND address
resolution messaging. The Client sends the NS(AR) message to the
Proxy/Server with an OMNI option with a HIP "Update/Sequence" message
sub-option. The Client sets the NS source address to the Client's
MNP-LLA and destination address to the target solicited node
multicast address. The Client wraps the NS message in an OAL header
with source address set to its own MNP-ULA and destination address
set to the Proxy/Server's ADM-ULA. The Client then performs OAL
fragmentation if necessary, wraps each fragment in a UDP/IP header
and sends it to the Proxy/Server.

When the Proxy/Server receives the OAL-encapsulated NS, it performs
OAL reassembly if necessary. The Proxy/Server then authenticates the
message by processing the HIP message sub-option and sends a
corresponding NS(AR) message over the spanning tree the same as if it
were the ROS, but with the OAL source address set to the Proxy/
Server's ADM-ULA, with destination set to the MNP-ULA of the target,
and with an OMNI option that includes no sub-options. When the ROR
receives the NS(AR), it adds the Proxy/Server's ADM-ULA and Client's
MNP-LLA to the target's Report List, and returns an NA(AR) with OMNI
option information for the target including all of the target's
Interface Attributes. The ROR sets the NA(AR) source address to the

target's MNP-LLA and sets the destination address to the Client's
MNP-LLA, then sets the OAL source address to the ADM-ULA of the ROR
and the destination to the ADM-ULA of the Proxy/Server. When the
Proxy/Server receives the NA(AR) message, it rewrites the OAL source
address to its own ADM-ULA and the destination address to the MNP-ULA
of the Client, then includes a HIP "Update/Acknowledge" message sub-
option in the OMNI option, performs OAL fragmentation if necessary,
wraps each fragment in UDP/IP headers, and sends it to the Client.

Following route optimization for targets in the same OMNI link
segment, if the target's L2ADDR is on the open INET, the Client
forwards carrier packets directly to the target INET address. If the
target is behind a NAT, the Client first establishes NAT state for
the L2ADDR using the "direct bubble" and NUD mechanisms discussed in
Section 3.10.1. The Client continues to send carrier packets via its
Proxy/Server until NAT state is populated, then begins forwarding
carrier packets via the direct path through the NAT to the target.
For targets in different OMNI link segments, the Client uses OAL/ORH
encapsulation and forwards carrier packets to the Bridge that
returned the NA message.

The ROR may return uNAs via the Proxy/Server if the target moves, and
the Proxy/Server will send corresponding uNAs to the Client with a
HIP "Notify" authentication message. The Client can also send NUD
messages to test forward path reachability even though there is no
security association between the Client and the target.

The Client can send original IP packets to route-optimized neighbors
in the same OMNI link segment no larger than the path MPS in one
piece and without OAL encapsulation and fragmentation, but with *NET
encapsulation. For larger original IP packets, the Client applies
OAL encapsulation and fragmentation if necessary according to
Section 3.9, with OAL header with source set to its own MNP-ULA and
destination set to the MNP-ULA of the target. The Client then
encapsulates each original IP packet or OAL fragment in UDP/IP *NET
headers and sends them to the next hop.

Note: The NAT traversal procedures specified in this document are
applicable for Cone, Address-Restricted and Port-Restricted NATs
only. While future updates to this document may specify procedures
for other NAT variations (e.g., hairpinning and various forms of
Symmetric NATs), it should be noted that continuous communications
are always possible through forwarding via a Proxy/Server even if NAT
traversal is not employed.

Note: Following the initial HIP Initiator/Responder exchange, AERO
Clients with OMNI interfaces configured over the open Internet
maintain HIP associations through the transmission of IPv6 ND

messages that include OMNI options with HIP "Update" and "Notify"
messages. OMNI interfaces use the HIP "Update" message when an
acknowledgement is required, and use the "Notify" message in
unacknowledged isolated IPv6 ND messages (e.g., unsolicited NAs).

Note: Proxy/Servers on the open Internet that act as Proxys
authenticate and remove OMNI option HIP message sub-options from RSes
they forward from the MN to another Proxy/Server, and insert and sign
HIP message and Origin Indication sub-options in RAs they forward
from another Proxy/Server to the MN. Conversely, Proxy/Servers that
act as Proxys forward without processing any MNP registration/
delegation information in RS/RA message exchanges between MNs and
other Proxy/Servers. The Proxy/Server acting as a Proxy is therefore
responsible for MN authentication, while the other Proxy/Servers are
responsible for registering/delegating MNPs (noting that the same
node can act as both Proxy and Proxy/Server).

3.23. 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 MN may be willing to
sacrifice a modicum of efficiency in order to have time-varying MNPs
that can be changed every so often to defeat adversarial tracking.

The DHCPv6 service offers a way for Clients that desire time-varying
MNPs to obtain short-lived prefixes (e.g., on the order of a small
number of minutes). In that case, the identity of the Client would
not be bound to the MNP but rather to a Node Identification value
(see: [I-D.templin-6man-omni-interface]) to be used as the Client ID
seed for MNP prefix delegation. The Client would then be obligated
to renumber its internal networks whenever its MNP (and therefore
also its MNP-LLA) changes. This should not present a challenge for
Clients with automated network renumbering services, however presents
limits for the durations of ongoing sessions that would prefer to use
a constant address.

4. Implementation Status

An early AERO implementation based on OpenVPN (https://openvpn.net/)
was announced on the v6ops mailing list on January 10, 2018 and an
initial public release of the AERO proof-of-concept source code was
announced on the intarea mailing list on August 21, 2015.

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

5. IANA Considerations

   The IANA is instructed to assign a new type value TBD1 in the IPv6
   Routing Types registry.

   The IANA has assigned the UDP port number "8060" for an earlier
   experimental first version of AERO [RFC6706]. This document
   obsoletes [RFC6706], and together with
   [I-D.templin-6man-omni-interface] 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 the invalid ICMPv6 message type number '0'.)
   This document makes no request of IANA, since
   [I-D.templin-6man-omni-interface] already provides instructions.

   No further IANA actions are required.

6. Security Considerations

   AERO Bridges configure secured tunnels with AERO Proxy/Servers,
   Relays and Proxys within their local OMNI link segments. Applicable
   secured tunnel alternatives include IPsec [RFC4301], TLS/SSL
   [RFC8446], DTLS [RFC6347], WireGuard [WG], etc. The AERO Bridges of
   all OMNI link segments in turn configure secured tunnels for their
   neighboring AERO Bridges in a spanning tree topology. Therefore,
   control messages exchanged between any pair of OMNI link neighbors on
   the spanning tree are already secured.

   AERO nodes acting as Route Optimization Responders (RORs) may also
   receive packets directly from arbitrary nodes in INET partitions
   instead of via the spanning tree. For INET partitions that apply
   effective ingress filtering to defeat source address spoofing, the
   simple data origin authentication procedures in Section 3.8 can be
   applied.

   For INET partitions that require strong security in the data plane,
   two options for securing communications include 1) disable route
   optimization so that all traffic is conveyed over secured tunnels, or
   2) enable on-demand secure tunnel creation between INET partition
   neighbors. Option 1) would result in longer routes than necessary
   and traffic concentration on critical infrastructure elements.
   Option 2) could be coordinated by establishing a secured tunnel on-
   demand instead of performing an NS/NA exchange in the route

optimization procedures. Procedures for establishing on-demand
secured tunnels are out of scope.

AERO Clients that connect to secured ANETs need not apply security to
their ND messages, since the messages will be intercepted by a
perimeter Proxy that applies security on its INET-facing interface as
part of the spanning tree (see above). AERO Clients connected to the
open INET can use network and/or transport layer security 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
authentication services specified in [RFC7401] and/or [RFC4380]
should be applied.

Application endpoints SHOULD use application-layer security services
such as TLS/SSL, DTLS or SSH [RFC4251] to assure the same level of
protection as for critical secured Internet services. AERO Clients
that require host-based VPN services SHOULD use network and/or
transport layer security services such as IPsec, TLS/SSL, DTLS, etc.
AERO Proxys and Proxy/Servers can also provide a network-based VPN
service on behalf of the Client, e.g., if the Client is located
within a secured enclave and cannot establish a VPN on its own
behalf.

AERO Proxy/Servers and Bridges present targets for traffic
amplification Denial of Service (DoS) attacks. This concern is no
different than for widely-deployed VPN security gateways in the
Internet, where attackers could send spoofed packets to the gateways
at high data rates. This can be mitigated by connecting Proxy/
Servers and Bridges over dedicated links with no connections to the
Internet and/or when connections to the Internet are only permitted
through well-managed firewalls. Traffic amplification DoS attacks
can also target an AERO Client's low data rate links. This is a
concern not only for Clients located on the open Internet but also
for Clients in secured enclaves. AERO Proxy/Servers and Proxys can
institute rate limits that protect Clients from receiving packet
floods that could DoS low data rate links.

AERO Relays must implement ingress filtering to avoid a spoofing
attack in which spurious messages with ULA addresses are injected
into an OMNI link from an outside attacker. AERO Clients MUST ensure
that their connectivity is not used by unauthorized nodes on their
EUNs to gain access to a protected network, i.e., AERO Clients that
act as routers MUST NOT provide routing services for unauthorized
nodes. (This concern is no different than for ordinary hosts that
receive an IP address delegation but then "share" the address with
other nodes via some form of Internet connection sharing such as
tethering.)

The MAP list MUST be well-managed and secured from unauthorized
tampering, even though the list contains only public information.
The MAP list can be conveyed to the Client in a similar fashion as in
[RFC5214] (e.g., through layer 2 data link login messaging, secure
upload of a static file, DNS lookups, etc.).

SRH authentication facilities are specified in [RFC8754].

Security considerations for accepting link-layer ICMP messages and
reflected packets are discussed throughout the document.

Security considerations for IPv6 fragmentation and reassembly are
discussed in [I-D.templin-6man-omni-interface].

7. Acknowledgements

Discussions in the IETF, aviation standards communities and private
exchanges helped shape some of the concepts in this work.
Individuals who contributed insights include Mikael Abrahamsson, Mark
Andrews, Fred Baker, Bob Braden, Stewart Bryant, Brian Carpenter,
Wojciech Dec, Pavel Drasil, Ralph Droms, Adrian Farrel, Nick Green,
Sri Gundavelli, Brian Haberman, Bernhard Haindl, Joel Halpern, Tom
Herbert, Sascha Hlusiak, Lee Howard, Zdenek Jaron, Andre Kostur,
Hubert Kuenig, Ted Lemon, Andy Malis, Satoru Matsushima, Tomek
Mrugalski, Madhu Niraula, Alexandru Petrescu, Behcet Saikaya, Michal
Skorepa, Joe Touch, Bernie Volz, Ryuji Wakikawa, Tony Whyman, Lloyd
Wood and James Woodyatt. Members of the IESG also provided valuable
input during their review process that greatly improved the document.
Special thanks go to Stewart Bryant, Joel Halpern and Brian Haberman
for their shepherding guidance during the publication of the AERO
first edition.

This work has further been encouraged and supported by Boeing
colleagues including Kyle Bae, M. Wayne Benson, Dave Bernhardt, Cam
Brodie, John Bush, Balaguruna Chidambaram, Irene Chin, Bruce Cornish,
Claudiu Danilov, Don Dillenburg, Joe Dudkowski, Wen Fang, Samad
Farooqui, Anthony Gregory, Jeff Holland, Seth Jahne, Brian Jaury,
Greg Kimberly, Ed King, Madhuri Madhava Badgandi, Laurel Matthew,
Gene MacLean III, Rob Muszkiewicz, Sean O'Sullivan, Vijay
Rajagopalan, Greg Saccone, Rod Santiago, Kent Shuey, Brian Skeen,
Mike Slane, Carrie Spiker, Katie Tran, Brendan Williams, Amelia
Wilson, Julie Wulff, Yueli Yang, Eric Yeh and other members of the
Boeing mobility, networking and autonomy teams. Kyle Bae, Wayne
Benson, Katie Tran and Eric Yeh are especially acknowledged for
implementing the AERO functions as extensions to the public domain
OpenVPN distribution.

Earlier works on NBMA tunneling approaches are found in
[RFC2529][RFC5214][RFC5569].

Many of the constructs presented in this second edition of AERO are
based on the author's earlier works, including:

o The Internet Routing Overlay Network (IRON)
   [RFC6179][I-D.templin-ironbis]

o Virtual Enterprise Traversal (VET)
   [RFC5558][I-D.templin-intarea-vet]

o The Subnetwork Encapsulation and Adaptation Layer (SEAL)
   [RFC5320][I-D.templin-intarea-seal]

o AERO, First Edition [RFC6706]

Note that these works cite numerous earlier efforts that are not also
cited here due to space limitations. The authors of those earlier
works are acknowledged for their insights.

This work is aligned with the NASA Safe Autonomous Systems Operation
(SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the FAA as per the SE2025 contract number
DTFAWA-15-D-00030.

This work is aligned with the Boeing Commercial Airplanes (BCA)
Internet of Things (IoT) and autonomy programs.

This work is aligned with the Boeing Information Technology (BIT)
MobileNet program.

8. References

8.1. Normative References

[I-D.templin-6man-omni-interface]
         Templin, F. and T. Whyman, "Transmission of IP Packets
         over Overlay Multilink Network (OMNI) Interfaces", draft-
         templin-6man-omni-interface-69 (work in progress), January
         2021.

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

   [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5,
              RFC 792, DOI 10.17487/RFC0792, September 1981,
              <https://www.rfc-editor.org/info/rfc792>.

   [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>.

   [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>.

   [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>.

   [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)",
              RFC 3972, DOI 10.17487/RFC3972, March 2005,
              <https://www.rfc-editor.org/info/rfc3972>.

   [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>.

   [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>.

   [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>.

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

   [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>.

   [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>.

   [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>.

   [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>.

8.2. Informative References

   [BGP]   Huston, G., "BGP in 2015, http://potaroo.net", January
              2016.

   [I-D.bonica-6man-comp-rtg-hdr]
              Bonica, R., Kamite, Y., Alston, A., Henriques, D., and L.
              Jalil, "The IPv6 Compact Routing Header (CRH)", draft-
              bonica-6man-comp-rtg-hdr-24 (work in progress), January
              2021.

   [I-D.bonica-6man-crh-helper-opt]
              Li, X., Bao, C., Ruan, E., and R. Bonica, "Compressed
              Routing Header (CRH) Helper Option", draft-bonica-6man-
              crh-helper-opt-02 (work in progress), October 2020.

   [I-D.ietf-intarea-frag-fragile]
              Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O.,
              and F. Gont, "IP Fragmentation Considered Fragile", draft-
              ietf-intarea-frag-fragile-17 (work in progress), September
              2019.

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

   [I-D.ietf-ipwave-vehicular-networking]
              Jeong, J., "IPv6 Wireless Access in Vehicular Environments
              (IPWAVE): Problem Statement and Use Cases", draft-ietf-
              ipwave-vehicular-networking-19 (work in progress), July
              2020.

   [I-D.ietf-rtgwg-atn-bgp]
              Templin, F., Saccone, G., Dawra, G., Lindem, A., and V.
              Moreno, "A Simple BGP-based Mobile Routing System for the
              Aeronautical Telecommunications Network", draft-ietf-
              rtgwg-atn-bgp-10 (work in progress), January 2021.

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

   [I-D.templin-intarea-seal]
              Templin, F., "The Subnetwork Encapsulation and Adaptation
              Layer (SEAL)", draft-templin-intarea-seal-68 (work in
              progress), January 2014.

   [I-D.templin-intarea-vet]
              Templin, F., "Virtual Enterprise Traversal (VET)", draft-
              templin-intarea-vet-40 (work in progress), May 2013.

   [I-D.templin-ipwave-uam-its]
              Templin, F., "Urban Air Mobility Implications for
              Intelligent Transportation Systems", draft-templin-ipwave-
              uam-its-04 (work in progress), January 2021.

   [I-D.templin-ironbis]
              Templin, F., "The Interior Routing Overlay Network
              (IRON)", draft-templin-ironbis-16 (work in progress),
              March 2014.

   [I-D.templin-v6ops-pdhost]
              Templin, F., "IPv6 Prefix Delegation and Multi-Addressing
              Models", draft-templin-v6ops-pdhost-27 (work in progress),
              January 2021.

   [OVPN]   OpenVPN, O., "http://openvpn.net", October 2016.

   [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>.

   [RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers",
              RFC 1812, DOI 10.17487/RFC1812, June 1995,
              <https://www.rfc-editor.org/info/rfc1812>.

   [RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003,
              DOI 10.17487/RFC2003, October 1996,
              <https://www.rfc-editor.org/info/rfc2003>.

   [RFC2004] Perkins, C., "Minimal Encapsulation within IP", RFC 2004,
              DOI 10.17487/RFC2004, October 1996,
              <https://www.rfc-editor.org/info/rfc2004>.

   [RFC2236] Fenner, W., "Internet Group Management Protocol, Version
              2", RFC 2236, DOI 10.17487/RFC2236, November 1997,
              <https://www.rfc-editor.org/info/rfc2236>.

   [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>.

   [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>.

   [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>.

   [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>.

   [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>.

   [RFC4251] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH)
              Protocol Architecture", RFC 4251, DOI 10.17487/RFC4251,
              January 2006, <https://www.rfc-editor.org/info/rfc4251>.

   [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>.

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

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

   [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>.

   [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>.

   [RFC4511] Sermersheim, J., Ed., "Lightweight Directory Access
              Protocol (LDAP): The Protocol", RFC 4511,
              DOI 10.17487/RFC4511, June 2006,
              <https://www.rfc-editor.org/info/rfc4511>.

   [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>.

   [RFC4982] Bagnulo, M. and J. Arkko, "Support for Multiple Hash
              Algorithms in Cryptographically Generated Addresses
              (CGAs)", RFC 4982, DOI 10.17487/RFC4982, July 2007,
              <https://www.rfc-editor.org/info/rfc4982>.

   [RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano,
              "Bidirectional Protocol Independent Multicast (BIDIR-
              PIM)", RFC 5015, DOI 10.17487/RFC5015, October 2007,
              <https://www.rfc-editor.org/info/rfc5015>.

   [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>.

   [RFC5320] Templin, F., Ed., "The Subnetwork Encapsulation and
              Adaptation Layer (SEAL)", RFC 5320, DOI 10.17487/RFC5320,
              February 2010, <https://www.rfc-editor.org/info/rfc5320>.

   [RFC5522] Eddy, W., Ivancic, W., and T. Davis, "Network Mobility
              Route Optimization Requirements for Operational Use in
              Aeronautics and Space Exploration Mobile Networks",
              RFC 5522, DOI 10.17487/RFC5522, October 2009,
              <https://www.rfc-editor.org/info/rfc5522>.

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

   [RFC5569] Despres, R., "IPv6 Rapid Deployment on IPv4
              Infrastructures (6rd)", RFC 5569, DOI 10.17487/RFC5569,
              January 2010, <https://www.rfc-editor.org/info/rfc5569>.

   [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>.

   [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli,
              "IPv6 Router Advertisement Options for DNS Configuration",
              RFC 6106, DOI 10.17487/RFC6106, November 2010,
              <https://www.rfc-editor.org/info/rfc6106>.

   [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful
              NAT64: Network Address and Protocol Translation from IPv6
              Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146,
              April 2011, <https://www.rfc-editor.org/info/rfc6146>.

   [RFC6179] Templin, F., Ed., "The Internet Routing Overlay Network
              (IRON)", RFC 6179, DOI 10.17487/RFC6179, March 2011,
              <https://www.rfc-editor.org/info/rfc6179>.

   [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>.

   [RFC6273] Kukec, A., Krishnan, S., and S. Jiang, "The Secure
              Neighbor Discovery (SEND) Hash Threat Analysis", RFC 6273,
              DOI 10.17487/RFC6273, June 2011,
              <https://www.rfc-editor.org/info/rfc6273>.

   [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
              January 2012, <https://www.rfc-editor.org/info/rfc6347>.

   [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>.

   [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>.

   [RFC7333] Chan, H., Ed., Liu, D., Seite, P., Yokota, H., and J.
              Korhonen, "Requirements for Distributed Mobility
              Management", RFC 7333, DOI 10.17487/RFC7333, August 2014,
              <https://www.rfc-editor.org/info/rfc7333>.

[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>.

[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>.

[RFC8446]   Rescorla, E., "The Transport Layer Security (TLS) Protocol
            Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018,
            <https://www.rfc-editor.org/info/rfc8446>.

[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>.

[WG]   Wireguard, "Wireguard, https://www.wireguard.com", August
            2020.

Appendix A. Non-Normative Considerations

AERO can be applied to a multitude of Internetworking scenarios, with
each having its own adaptations. The following considerations are
provided as non-normative guidance:

A.1. Implementation Strategies for Route Optimization

Route optimization as discussed in Section 3.14 results in the route
optimization source (ROS) creating a neighbor cache entry for the
target neighbor. The neighbor cache entry is maintained for at most
ReachableTime seconds and then deleted unless updated. In order to
refresh the neighbor cache entry lifetime before the ReachableTime
timer expires, the specification requires implementations to issue a
new NS/NA exchange to reset ReachableTime while data packets are
still flowing. However, the decision of when to initiate a new NS/NA
exchange and to perpetuate the process is left as an implementation
detail.

One possible strategy may be to monitor the neighbor cache entry
watching for data packets for (ReachableTime - 5) seconds. If any
data packets have been sent to the neighbor within this timeframe,
then send an NS to receive a new NA. If no data packets have been
sent, wait for 5 additional seconds and send an immediate NS if any
data packets are sent within this "expiration pending" 5 second

window. If no additional data packets are sent within the 5 second
window, delete the neighbor cache entry.

The monitoring of the neighbor data packet traffic therefore becomes
an asymmetric ongoing process during the neighbor cache entry
lifetime. If the neighbor cache entry expires, future data packets
will trigger a new NS/NA exchange while the packets themselves are
delivered over a longer path until route optimization state is re-
established.

A.2. Implicit Mobility Management

OMNI interface neighbors MAY provide a configuration option that
allows them to perform implicit mobility management in which no ND
messaging is used. In that case, the Client only transmits packets
over a single interface at a time, and the neighbor always observes
packets arriving from the Client from the same link-layer source
address.

If the Client's underlying interface address changes (either due to a
readdressing of the original interface or switching to a new
interface) the neighbor immediately updates the neighbor cache entry
for the Client and begins accepting and sending packets according to
the Client's new address. This implicit mobility method applies to
use cases such as cellphones with both WiFi and Cellular interfaces
where only one of the interfaces is active at a given time, and the
Client automatically switches over to the backup interface if the
primary interface fails.

A.3. Direct Underlying Interfaces

When a Client's OMNI interface is configured over a Direct interface,
the neighbor at the other end of the Direct link can receive packets
without any encapsulation. In that case, the Client sends packets
over the Direct link according to QoS preferences. If the Direct
interface has the highest QoS preference, then the Client's IP
packets are transmitted directly to the peer without going through an
ANET/INET. If other interfaces have higher QoS preferences, then the
Client's IP packets are transmitted via a different interface, which
may result in the inclusion of Proxys, Proxy/Servers and Bridges in
the communications path. Direct interfaces must be tested
periodically for reachability, e.g., via NUD.

A.4. AERO Critical Infrastructure Considerations

AERO Bridges can be either Commercial off-the Shelf (COTS) standard
IP routers or virtual machines in the cloud. Bridges must be
provisioned, supported and managed by the INET administrative

authority, and connected to the Bridges of other INETs via inter-
domain peerings. Cost for purchasing, configuring and managing
Bridges is nominal even for very large OMNI links.

AERO Proxy/Servers can be standard dedicated server platforms, but
most often will be deployed as virtual machines in the cloud. The
only requirements for Proxy/Servers are that they can run the AERO
user-level code and have at least one network interface connection to
the INET. As with Bridges, Proxy/Servers must be provisioned,
supported and managed by the INET administrative authority. Cost for
purchasing, configuring and managing Proxy/Servers is nominal
especially for virtual Proxy/Servers hosted in the cloud.

AERO Proxys are most often standard dedicated server platforms with
one network interface connected to the ANET and a second interface
connected to an INET. As with Proxy/Servers, the only requirements
are that they can run the AERO user-level code and have at least one
interface connection to the INET. Proxys must be provisioned,
supported and managed by the ANET administrative authority. Cost for
purchasing, configuring and managing Proxys is nominal, and borne by
the ANET administrative authority.

AERO Relays can be any dedicated server or COTS router platform
connected to INETs and/or EUNs. The Relay connects to the OMNI link
and engages in eBGP peering with one or more Bridges as a stub AS.
The Relay then injects its MNPs and/or non-MNP prefixes into the BGP
routing system, and provisions the prefixes to its downstream-
attached networks. The Relay can perform ROS/ROR services the same
as for any Proxy/Server, and can route between the MNP and non-MNP
address spaces.

[A.5](#). **AERO Server Failure Implications**

AERO Proxy/Servers may appear as a single point of failure in the
architecture, but such is not the case since all Proxy/Servers on the
link provide identical services and loss of a Proxy/Server does not
imply immediate and/or comprehensive communication failures.
Although Clients typically associate with a single Proxy/Server at a
time, Proxy/Server failure is quickly detected and conveyed by
Bidirectional Forward Detection (BFD) and/or proactive NUD allowing
Clients to migrate to new Proxy/Servers.

If a Proxy/Server fails, ongoing packet forwarding to Clients will
continue by virtue of the neighbor cache entries that have already
been established in route optimization sources (ROSs). If a Client
also experiences mobility events at roughly the same time the Proxy/
Server fails, unsolicited NA messages may be lost but proxy neighbor
cache entries in the DEPARTED state will ensure that packet

   forwarding to the Client's new locations will continue for up to
   DepartTime seconds.

   If a Client is left without a Proxy/Server for an extended timeframe
   (e.g., greater than ReachableTime seconds) then existing neighbor
   cache entries will eventually expire and both ongoing and new
   communications will fail. The original source will continue to
   retransmit until the Client has established a new Proxy/Server
   relationship, after which time continuous communications will resume.

   Therefore, providing many Proxy/Servers on the link with high
   availability profiles provides resilience against loss of individual
   Proxy/Servers and assurance that Clients can establish new Proxy/
   Server relationships quickly in event of a Proxy/Server failure.

A.6. **AERO Client / Server Architecture**

   The AERO architectural model is client / server in the control plane,
   with route optimization in the data plane. The same as for common
   Internet services, the AERO Client discovers the addresses of AERO
   Proxy/Servers and selects one Proxy/Server to connect to. The AERO
   service is analogous to common Internet services such as google.com,
   yahoo.com, cnn.com, etc. However, there is only one AERO service for
   the link and all Proxy/Servers provide identical services.

   Common Internet services provide differing strategies for advertising
   server addresses to clients. The strategy is conveyed through the
   DNS resource records returned in response to name resolution queries.
   As of January 2020 Internet-based 'nslookup' services were used to
   determine the following:

   o When a client resolves the domainname "google.com", the DNS always
      returns one A record (i.e., an IPv4 address) and one AAAA record
      (i.e., an IPv6 address). The client receives the same addresses
      each time it resolves the domainname via the same DNS resolver,
      but may receive different addresses when it resolves the
      domainname via different DNS resolvers. But, in each case,
      exactly one A and one AAAA record are returned.

   o When a client resolves the domainname "ietf.org", the DNS always
      returns one A record and one AAAA record with the same addresses
      regardless of which DNS resolver is used.

   o When a client resolves the domainname "yahoo.com", the DNS always
      returns a list of 4 A records and 4 AAAA records. Each time the
      client resolves the domainname via the same DNS resolver, the same
      list of addresses are returned but in randomized order (i.e.,
      consistent with a DNS round-robin strategy). But, interestingly,

the same addresses are returned (albeit in randomized order) when
the domainname is resolved via different DNS resolvers.

o When a client resolves the domainname "amazon.com", the DNS always
   returns a list of 3 A records and no AAAA records. As with
   "yahoo.com", the same three A records are returned from any
   worldwide Internet connection point in randomized order.

The above example strategies show differing approaches to Internet
resilience and service distribution offered by major Internet
services. The Google approach exposes only a single IPv4 and a
single IPv6 address to clients. Clients can then select whichever IP
protocol version offers the best response, but will always use the
same IP address according to the current Internet connection point.
This means that the IP address offered by the network must lead to a
highly-available server and/or service distribution point. In other
words, resilience is predicated on high availability within the
network and with no client-initiated failovers expected (i.e., it is
all-or-nothing from the client's perspective). However, Google does
provide for worldwide distributed service distribution by virtue of
the fact that each Internet connection point responds with a
different IPv6 and IPv4 address. The IETF approach is like google
(all-or-nothing from the client's perspective), but provides only a
single IPv4 or IPv6 address on a worldwide basis. This means that
the addresses must be made highly-available at the network level with
no client failover possibility, and if there is any worldwide service
distribution it would need to be conducted by a network element that
is reached via the IP address acting as a service distribution point.

In contrast to the Google and IETF philosophies, Yahoo and Amazon
both provide clients with a (short) list of IP addresses with Yahoo
providing both IP protocol versions and Amazon as IPv4-only. The
order of the list is randomized with each name service query
response, with the effect of round-robin load balancing for service
distribution. With a short list of addresses, there is still
expectation that the network will implement high availability for
each address but in case any single address fails the client can
switch over to using a different address. The balance then becomes
one of function in the network vs function in the end system.

The same implications observed for common highly-available services
in the Internet apply also to the AERO client/server architecture.
When an AERO Client connects to one or more ANETs, it discovers one
or more AERO Proxy/Server addresses through the mechanisms discussed
in earlier sections. Each Proxy/Server address presumably leads to a
fault-tolerant clustering arrangement such as supported by Linux-HA,
Extended Virtual Synchrony or Paxos. Such an arrangement has
precedence in common Internet service deployments in lightweight

virtual machines without requiring expensive hardware deployment.
Similarly, common Internet service deployments set service IP
addresses on service distribution points that may relay requests to
many different servers.

For AERO, the expectation is that a combination of the Google/IETF
and Yahoo/Amazon philosophies would be employed. The AERO Client
connects to different ANET access points and can receive 1-2 Proxy/
Server ADM-LLAs at each point. It then selects one AERO Proxy/Server
address, and engages in RS/RA exchanges with the same Proxy/Server
from all ANET connections. The Client remains with this Proxy/Server
unless or until the Proxy/Server fails, in which case it can switch
over to an alternate Proxy/Server. The Client can likewise switch
over to a different Proxy/Server at any time if there is some reason
for it to do so. So, the AERO expectation is for a balance of
function in the network and end system, with fault tolerance and
resilience at both levels.

Appendix B. Change Log

<< RFC Editor - remove prior to publication >>

Changes from draft-templin-intarea-6706bis-61 to draft-templin-
intrea-6706bis-62:

o New sub-section on OMNI Neighbor Interface Attributes

Changes from draft-templin-intarea-6706bis-59 to draft-templin-
intrea-6706bis-60:

o Removed all references to S/TLLAO - all Interface Attributes are
   now maintained completely in the OMNI option.

Changes from draft-templin-intarea-6706bis-58 to draft-templin-
intrea-6706bis-59:

o The term "Relay" used in older draft versions is now "Bridge".
   "Relay" now refers to what was formally called: "Gateway".

o Fine-grained cleanup of Forwarding Algorithm; IPv6 ND message
   addressing; OMNI Prefix Lengths, etc.

Changes from draft-templin-intarea-6706bis-54 to draft-templin-
intrea-6706bis-55:

o Updates on Segment Routing and S/TLLAO contents.

o Various editorials and addressing cleanups.

Changes from draft-templin-intarea-6706bis-52 to draft-templin-intrea-6706bis-53:

o Normative reference to the OMNI spec, and remove portions that are
   already specified in OMNI.

o Renamed "AERO interface/link" to "OMIN interface/link" throughout
   the document.

o Truncated obsolete back section matter.

Author's Address

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org