            Asymmetric Extended Route Optimization (AERO)
                 draft-templin-intarea-6706bis-15.txt

Abstract

   This document specifies the operation of IP over tunnel virtual links
   using Asymmetric Extended Route Optimization (AERO). AERO interfaces
   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 services are employed for network admission and to
   manage the routing system. Multilink operation, mobility management,
   quality of service (QoS) signaling and route optimization are
   naturally supported through dynamic neighbor cache updates. Standard
   IP multicasting services are also supported. AERO is a widely-
   applicable tunneling solution especially well-suited to aviation
   services, mobile Virtual Private Networks (VPNs) and many other
   applications.

Status of This Memo

Table of Contents

1. Introduction

   Asymmetric 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. AERO is based on a Non-Broadcast,

Multiple Access (NBMA) virtual link model known as the AERO link.
The AERO link is configured over one or more underlying
Internetworks, and nodes on the link can exchange IP packets via
tunneling. Multilink operation allows for increased reliability,
bandwidth optimization and traffic path diversity.

The AERO service comprises Clients, Proxys, Servers, and Gateways
that are seen as AERO link neighbors. Each node's AERO interface
uses an IPv6 link-local address format (known as the AERO address)
that supports operation of the IPv6 Neighbor Discovery (ND) protocol
[RFC4861] and links ND to IP forwarding. A node's AERO interface can
be configured over multiple underlying interfaces, and may therefore
may appear 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 links provide a cloud-based service where mobile nodes may use
any Server acting as a Mobility Anchor Point (MAP) and fixed nodes
may use any Gateway on the link for efficient communications. Fixed
nodes forward packets destined to other AERO nodes to the nearest
Gateway, which forwards them through the cloud. A mobile node's
initial packets are forwarded through the MAP, while direct routing
is supported through asymmetric route optimization while data 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 Relays are interconnected in a secured private BGP overlay
routing instance known as the "SPAN". The SPAN provides a hybrid
routing/bridging service to join the underlying Internetworks of
multiple disjoint administrative domains into a single unified AERO
link. Each AERO link instance is characterized by the set of
Mobility Service Prefixes (MSPs) common to all mobile nodes. The
link should extend to the point where a Gateway/MAP is on the optimal
route from any correspondent node on the link, and provides a gateway
between the underlying Internetwork and the SPAN. To the underlying
Internetwork, the Gateway/MAP is the source of a route to its MSP,
and hence uplink traffic to the mobile node is naturally routed to
the nearest Gateway/MAP.

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 Virtual Private Network (VPN) links of
mobile nodes (e.g., cellphones, tablets, laptop computers, etc.) that
connect into a home enterprise network via public access networks
using services such as OpenVPN [OVPN]. 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; 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
   interfaces use the ND service specified in [RFC4861].

IPv6 Prefix Delegation (PD)
   a networking service for delegating IPv6 prefixes to nodes on the
   link. The nominal PD service is DHCPv6 [RFC8415], however
   alternate services (e.g., based on ND messaging) are also in scope
   [I-D.templin-v6ops-pdhost][I-D.templin-6man-dhcpv6-ndopt]. Most
   notably, a form of PD known as "Prefix Assertion" can be used if
   the prefix can be represented 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, or the public Internet itself. For secured ANETs, link-
   layer security services such as IEEE 802.1X and physical-layer
   security prevent unauthorized access internally while border
   network-layer security services such as firewalls and proxies
   prevent unauthorized outside access.

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

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

Internetwork (INET)
   a connected IP network topology with a coherent routing and
   addressing plan and that provides an Internetworking backbone
   service. 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 Partition
   frequently, INETs such as large corporate enterprise networks are
   sub-divided internally into separate isolated partitions. Each
   partition is fully connected internally but disconnected form
   other partitions, and there is no requirement that separate
   partitions maintain consistent Internet Protocol and/or addressing
   plans. (An INET partition is the same as a SPAN segment discussed
   below.)

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

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

AERO link
   a Non-Broadcast, Multiple Access (NBMA) tunnel virtual overlay
   configured over one or more underlying INETs. Nodes on the AERO
   link appear as single-hop neighbors from the perspective of the
   virtual overlay even though they may be separated by many
   underlying INET hops. AERO links may be configured over multiple
   underlying SPAN segments (see below).

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

AERO address
   an IPv6 link-local address assigned to an AERO interface and
   constructed as specified in Section 3.4.

base AERO address
   the lowest-numbered AERO address aggregated by the MNP (see
   Section 3.4).

Mobility Service Prefix (MSP)
   an IP prefix assigned to the AERO link and from which more-
   specific Mobile Network Prefixes (MNPs) are derived.

Mobile Network Prefix (MNP)
   an IP prefix allocated from an MSP and delegated to an AERO Client
   or Gateway.

AERO node
   a node that is connected to an AERO link, or that provides
   services to other nodes on an AERO link.

AERO Client ("Client")
   an AERO node that connects to one or more ANETs and requests MNP
   PDs from AERO Servers. Following PD, the Client assigns a Client
   AERO address to the AERO interface for use in ND exchanges with
   other AERO nodes and forwards packets to correspondents according
   to AERO interface neighbor cache state.

AERO Server ("Server")
   an INET node that configures an AERO interface to provide default
   forwarding services and a Mobility Anchor Point (MAP) for AERO
   Clients. The Server assigns an administratively-provisioned AERO
   address to the AERO interface to support the operation of the ND/
   PD services, and advertises all of its associated MNPs via BGP
   peerings with Relays.

AERO Gateway ("Gateway")
   an AERO Server that also provides forwarding services between
   nodes reached via the AERO link and correspondents on other links.
   AERO Gateways are provisioned with MNPs (i.e., the same as for an
   AERO Client) and run a dynamic routing protocol to discover any
   non-MNP IP routes. In both cases, the Gateway advertises the
   MSP(s) over INET interfaces, and distributes all of its associated
   MNPs and non-MNP IP routes via BGP peerings with Relays (i.e., the
   same as for an AERO Server).

AERO Relay ("Relay")
   a node that provides hybrid routing/bridging services (as well as
   a security trust anchor) for nodes on an AERO link. As a router,
   the Relay forwards packets using standard IP forwarding. As a
   bridge, the Relay forwards packets over the AERO link without
   decrementing the IPv6 Hop Limit. AERO Relays peer with Servers
   and other Relays to discover the full set of MNPs for the link as
   well as any non-MNPs that are reachable via Gateways.

AERO Proxy ("Proxy")

    a node that provides proxying services between Clients in an ANET
    and Servers in external INETs. The AERO Proxy is a conduit
    between the ANET and external INETs in the same manner as for
    common web proxies, and behaves in a similar fashion as for ND
    proxies [RFC4389].

  Spanning Partitioned AERO Networks (SPAN)
    a means for bridging disjoint INET partitions as segments of a
    unified AERO link the same as for a bridged campus LAN. The SPAN
    is a mid-layer IPv6 encapsulation service in the AERO routing
    system that supports a unified AERO link view for all segments.
    Each segment in the SPAN is a distinct INET partition.

  SPAN Service Prefix (SSP)
    a global or unique local /96 IPv6 prefix assigned to the AERO link
    to support SPAN services.

  SPAN Partition Prefix (SPP)
    a sub-prefix of the SPAN Service Prefix uniquely assigned to a
    single SPAN segment.

  SPAN Address
    a global or unique local IPv6 address taken from a SPAN Partition
    Prefix and constructed as specified in Section 3.5. SPAN
    addresses are statelessly derived from AERO addresses, and vice-
    versa.

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

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

  link-layer address
    an IP address used as an encapsulation header source or
    destination address from the perspective of the AERO interface.
    When UDP encapsulation is used, the UDP port number is also
    considered as part of the link-layer address. From the
    perspective of the AERO interface, the link-layer address is
    either an INET address for intra-segment encapsulation or a SPAN
    address for inter-segment encapsulation.

  network layer address
    the source or destination address of an encapsulated IP packet
    presented to the AERO interface.

end user network (EUN)
   an internal virtual or external edge IP network that an AERO
   Client or Gateway connects to the rest of the network via the AERO
   interface. The Client/Gateway sees each EUN as a "downstream"
   network, and sees the AERO 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 packets between any
   downstream-attached networks and the AERO link.

Mobility Anchor Point (MAP)
   an AERO Server that is currently tracking and reporting the
   mobility events of its associated Mobile Node Clients.

Route Optimization Source (ROS)
   the AERO node nearest the source that initiates route
   optimization. The ROS may be a Server or Proxy acting on behalf
   of the source Client.

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

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

ROS List
   a list of AERO/SPAN-to-INET address mappings of all ROSes within
   the same SPAN segment. There is a distinct ROS list for each
   segment.

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

Throughout the document, the simple terms "Client", "Server",
"Relay", "Proxy" and "Gateway" refer to "AERO Client", "AERO Server",
"AERO Relay", "AERO Proxy" and "AERO Gateway", 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. 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", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in [RFC2119]. Lower case
uses of these words are not to be interpreted as carrying RFC2119
significance.

3. Asymmetric Extended Route Optimization (AERO)

The following sections specify the operation of IP over Asymmetric
Extended Route Optimization (AERO) links:

3.1. AERO Link Reference Model

```
                        +----------------+
                        | AERO Relay R1  |
                        | Nbr: S1, S2, P1|
                        |(X1->S1; X2->S2)|
                        |      MSP M1    |
                        +-+---------+--+-+
      +--------------+     | Secured |  |     +--------------+
      |AERO Server S1|     | tunnels |  |     |AERO Server S2|
      |  Nbr: C1, R1 +-----+         |  +-----+  Nbr: C2, R1 |
      |  default->R1 |              |          | default->R1 |
      |     X1->C1   |              |          |    X2->C2   |
      +-------+------+              |          +------+-------+
              |      AERO Link      |                 |
      X---+---+------------------+--)--------------+---+---X
          |                      | |               |
      +-----+--------+   +--------+--+-----+   +--------+-----+
      |AERO Client C1|   |  AERO Proxy P1  |   |AERO Client C2|
      |    Nbr: S1   |   |(Proxy Nbr Cache)|   |    Nbr: S2   |
      | default->S1  |   +--------+--------+   | default->S2  |
      |    MNP X1    |            |            |    MNP X2    |
      +------+-------+   .--------+------.     +-----+--------+
             |          (- Proxyed Clients -)       |
           .-.          `---------------'         .-.
         ,-(  _)-.                              ,-(  _)-.
       .-(_  IP   )-.   +-------+   +-------+  .-(_  IP   )-.
      (__   EUN     )--|Host H1|   |Host H2|--(__   EUN     )
        `-(_____)-'    +-------+   +-------+    `-(_____)-'
```
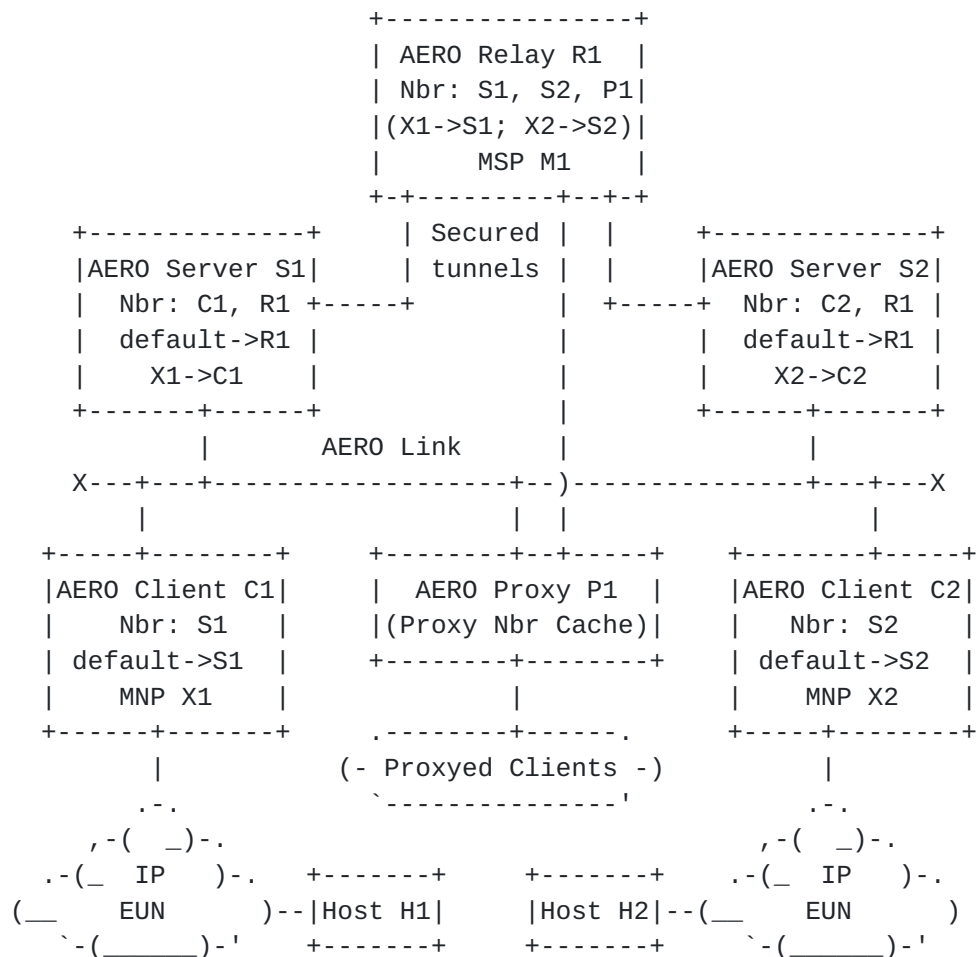
                    Figure 1: AERO Link Reference Model

   Figure 1 presents the AERO link reference model. In this model:

   o the AERO link is an overlay network service configured over one or
      more underlying INET partitions which may be managed by different
      administrative authorities and have incompatible protocols and/or
      addressing plans.

   o AERO Relay R1 aggregates Mobility Service Prefix (MSP) M1,
      discovers Mobile Network Prefixes (MNPs) X* and advertises the MSP
      via BGP peerings over secured tunnels to Servers (S1, S2). Relays
      use the SPAN service to bridge disjoint segments of a partitioned
      AERO link.

   o AERO Servers S1 and S2 configure secured tunnels with Relay R1 and
      also act as Mobility Anchor Points (MAPs) and default routers for
      their associated Clients C1 and C2.

o AERO Clients C1 and C2 associate with 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.

o AERO Proxy P1 configures a secured tunnel with Relay R1 and
   provides proxy services for AERO Clients in secured enclaves that
   cannot associate directly with other AERO link neighbors.

Each node on the AERO link maintains an AERO interface neighbor cache
and an IP forwarding table the same as for any link. Although the
figure shows a limited deployment, in common operational practice
there will normally be many additional Relays, Servers, Clients and
Proxys.

3.2. AERO Node Types

AERO Relays provide hybrid routing/bridging services (as well as a
security trust anchor) for nodes on an AERO link. Relays use
standard IPv6 routing to forward packets both within the same INET
partitions and between disjoint INET partitions based on a mid-layer
IPv6 encapsulation known as the SPAN header. The inner IP layer
experiences a virtual bridging service since the inner IP TTL/Hop
Limit is not decremented during forwarding. Each Relay also peers
with Servers and other Relays in a dynamic routing protocol instance
to provide a Distributed Mobility Management (DMM) service for the
list of active MNPs (see Section 3.3). Relays present the AERO link
as a set of one or more Mobility Service Prefixes (MSPs) but as link-
layer devices need not connect directly to the AERO link themselves
unless an administrative interface is desired. Relays configure
secured tunnels with Servers, Proxys and other Relays; they further
maintain IP forwarding table entries for each Mobile Network Prefix
(MNP) and any other reachable non-MNP prefixes.

AERO Servers provide default forwarding services and a Mobility
Anchor Point (MAP) for AERO Client Mobile Nodes (MNs). Each Server
also peers with Relays in a dynamic routing protocol instance to
advertise its list of associated MNPs (see Section 3.3). Servers
facilitate PD exchanges with Clients, where each delegated prefix
becomes an MNP taken from an MSP. Servers forward packets between
AERO interface neighbors and track each Client's mobility profiles.

AERO Clients receive MNPs through PD exchanges with AERO Servers over
the AERO link, and distribute the MNPs to nodes on EUNs. Each Client
can associate with a single Server or with multiple Servers (e.g.,
for fault tolerance, load balancing, etc). A Client may also be co-
resident on the same physical or virtual platform as a Server; in

that case, the Client and Server behave as a single functional unit
and without the need for any Client/Server control messaging.

AERO Proxys provide a conduit for AERO Clients in ANETs to associate
with AERO Servers in external INETs. Client and Servers exchange
control plane messages via the Proxy, which intercepts them at the
ANET/INET boundary. The Proxy forwards data packets to and from
Clients according to forwarding information in the neighbor cache.
The Proxy function is specified in Section 3.16.

AERO Gateways are Servers that provide forwarding services between
the AERO interface and INET/EUN interfaces. Gateways 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 Gateway
advertises the MSP(s) to INETs, and distributes all of its associated
MNPs and non-MNP IP routes via BGP peerings with Relays.

AERO Relays, Servers, Proxys and Gateways are critical infrastructure
elements in fixed (i.e., non-mobile) INET deployments and hence have
permanent and unchanging INET addresses. AERO Clients are MNs that
connect via ANET interfaces, i.e., their ANET addresses may change
when the Client moves to a new ANET connection.

3.3. AERO Routing System

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

In a reference deployment, each Server is configured as an Autonomous
System Border Router (ASBR) for a stub Autonomous System (AS) using
an AS Number (ASN) that is unique within the BGP instance, and each
Server further uses eBGP to peer with one or more Relays but does not
peer with other Servers. Each INET of a multi-segment AERO link must
include one or more Relays, which peer with the Servers and Proxys
within that INET. All Relays within the same INET are members of the
same hub AS using a common ASN, and use iBGP to maintain a consistent
view of all active MNPs currently in service. The Relays of
different INETs peer with one another using eBGP.

Relays advertise the AERO link's MSPs and any non-MNP routes to each
of their Servers. This means that any aggregated non-MNPs (including
"default") are advertised to all Servers. Each Relay configures a
black-hole route for each of its MSPs. By black-holing the MSPs, the
Relay will maintain forwarding table entries only for the MNPs that
are currently active, and packets destined to all other MNPs will
correctly incur Destination Unreachable messages due to the black-

hole route. In this way, Servers have only partial topology
knowledge (i.e., they know only about the MNPs of their directly
associated Clients) and they forward all other packets to Relays
which have full topology knowledge.

Servers maintain a working set of associated MNPs, and dynamically
announce new MNPs and withdraw departed MNPs in eBGP updates to
Relays. Servers that are configured as Gateways also redistribute
non-MNP routes learned from non-AERO interfaces via their eBGP Relay
peerings.

Clients are expected to remain associated with their current Servers
for extended timeframes, however Servers SHOULD selectively suppress
updates for impatient Clients that repeatedly associate and
disassociate with them in order to dampen routing churn. Servers
that are configured as Gateways advertise the MSPs via INET/EUN
interfaces, and forward packets between INET/EUN interfaces and the
AERO interface using standard IP forwarding.

For IPv6 MNPs, the AERO routing system includes only IPv6 routes.
For IPv4 MNPs, the AERO routing system includes both IPv4 routes and
also IPv6 routes based on the IPv4-mapped IPv6 address corresponding
to the MNP and with prefix length set to 96 plus the length of the
IPv4 prefix. (For example, if the IPv4 MNP is 192.0.2.0/24 then the
IPv4-mapped prefix is 0:0:0:0:0:FFFF:192.0.2.0/120.)

Scaling properties of the AERO routing system are limited by the
number of BGP routes that can be carried by Relays. As of 2015, the
global public Internet BGP routing system manages more than 500K
routes with linear growth and no signs of router resource exhaustion
[BGP]. More recent network emulation studies have also shown that a
single Relay can accommodate at least 1M dynamically changing BGP
routes even on a lightweight virtual machine, i.e., and without
requiring high-end dedicated router hardware.

Therefore, assuming each Relay can carry 1M or more routes, this
means that at least 1M Clients can be serviced by a single set of
Relays. A means of increasing scaling would be to assign a different
set of Relays for each set of MSPs. In that case, each Server still
peers with one or more Relays, but institutes route filters so that
BGP updates are only sent to the specific set of Relays that
aggregate the MSP. For example, if the MSP for the AERO link is
2001:db8::/32, a first set of Relays could service the MSP
2001:db8::/40, a second set of Relays could service
2001:db8:0100::/40, a third set could service 2001:db8:0200::/40,
etc.

Assuming up to 1K sets of Relays, the AERO routing system can then
accommodate 1B or more MNPs with no additional overhead (for example,
it should be possible to service 1B /64 MNPs taken from a /34 MSP and
even more for shorter prefixes). In this way, each set of Relays
services a specific set of MSPs that they advertise to the native
Internetwork routing system, and each Server configures MSP-specific
routes that list the correct set of Relays as next hops. This
arrangement also allows for natural incremental deployment, and can
support small scale initial deployments followed by dynamic
deployment of additional Clients, Servers and Relays without
disturbing the already-deployed base.

A full discussion of the BGP-based routing system used by AERO is
found in [I-D.ietf-rtgwg-atn-bgp]. The system provides for
Distributed Mobility Management (DMM) per the distributed mobility
anchoring architecture [I-D.ietf-dmm-distributed-mobility-anchoring].

3.4. AERO Addresses

A Client's AERO address is an IPv6 link-local address with an
interface identifier based on the Client's delegated MNP. Relay,
Server and Proxy AERO addresses are assigned from the range fe80::/96
and include an administratively-provisioned value in the lower 32
bits.

For IPv6, Client AERO addresses begin with the prefix fe80::/64 and
include in the interface identifier (i.e., the lower 64 bits) a
64-bit prefix taken from one of the Client's IPv6 MNPs. For example,
if the AERO Client receives the IPv6 MNP:

    2001:db8:1000:2000::/56

it constructs its corresponding AERO addresses as:

    fe80::2001:db8:1000:2000

    fe80::2001:db8:1000:2001

    fe80::2001:db8:1000:2002

    ... etc. ...

    fe80::2001:db8:1000:20ff

For IPv4, Client AERO addresses are based on an IPv4-mapped IPv6
address formed from an IPv4 MNP and with a Prefix Length of 96 plus
the MNP prefix length. For example, for the IPv4 MNP 192.0.2.32/28
the IPv4-mapped IPv6 MNP is:

      0:0:0:0:0:FFFF:192.0.2.16/124

   The Client then constructs its AERO addresses with the prefix
   fe80::/64 and with the lower 64 bits of the IPv4-mapped IPv6 address
   in the interface identifier as:

      fe80::FFFF:192.0.2.16

      fe80::FFFF:192.0.2.17

      fe80::FFFF:192.0.2.18

      ... etc. ...

      fe80:FFFF:192.0.2.31

   Relay, Server and Proxy AERO addresses are allocated from the range
   fe80::/96, and MUST be managed for uniqueness. The lower 32 bits of
   the AERO address includes a unique integer value (e.g., fe80::1,
   fe80::2, fe80::3, etc.) as assigned by the administrative authority
   for the link. If the link spans multiple SPAN segments, the AERO
   addresses are assigned to each segment in 1x1 correspondence with
   SPAN addresses (see: Section 3.5). The address fe80:: is reserved as
   the IPv6 link-local Subnet Router Anycast address [RFC4291], and the
   address fe80::ffff:ffff is reserved as the unspecified AERO address;
   hence, these values are not available general assignment.

   The lowest-numbered AERO address from a Client's MNP delegation
   serves as the "base" AERO address (for example, for the MNP
   2001:db8:1000:2000::/56 the base AERO address is
   fe80::2001:db8:1000:2000). The Client then assigns the base AERO
   address to the AERO interface and uses it for the purpose of
   maintaining the neighbor cache entry. The Server likewise uses the
   AERO address as its index into the neighbor cache for this Client.

   If the Client has multiple AERO addresses (i.e., when there are
   multiple MNPs and/or MNPs with prefix lengths shorter than /64), the
   Client originates ND messages using the base AERO address as the
   source address and accepts and responds to ND messages destined to
   any of its AERO addresses as equivalent to the base AERO address. In
   this way, the Client maintains a single neighbor cache entry that may
   be indexed by multiple AERO addresses.

   The Client's Subnet Router Anycast address can be statelessly
   determined from its AERO address by simply transposing the AERO
   address into the upper N bits of the Anycast address followed by
   128-N bits of zeros. For example, for the AERO address

   fe80::2001:db8:1:2 the subnet router anycast address is
   2001:db8:1:2::/64.

   AERO addresses for mobile node Clients embed a MNP as discussed
   above, while AERO addresses for non-MNP destinations are constructed
   in exactly the same way. A Client AERO address therefore encodes
   either an MNP if the prefix is reached via the SPAN or a non-MNP if
   the prefix is reached via a Gateway.

3.5. Spanning Partitioned AERO Networks (SPAN)

   In the simplest case, an AERO link configured over a single INET
   appears as a single unified link with a consistent underlying network
   addressing plan. In that case, all nodes on the link can exchange
   packets via encapsulation with INET addresses, since the underlying
   INET is connected. In common practice, however, an AERO link may be
   partitioned into multiple "segments", where each segment is a
   distinct INET potentially managed under a different administrative
   authority (e.g., as for worldwide aviation service providers such as
   ARINC, SITA, Inmarsat, etc.). Individual INETs may 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, proxies, packet filtering
   gateways, etc.), and in many cases disjoint segments may not even
   have any common physical link connections at all. Therefore, nodes
   can only be assured of exchanging 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 Relays.

   The same as for traditional campus LANs, multiple AERO link segments
   can be joined into a single unified link via a virtual bridging
   service termed the "SPAN". The SPAN performs link-layer packet
   forwarding between segments (i.e., bridging) without decrementing the
   network-layer TTL/Hop Limit. The SPAN model is depicted in Figure 2:

```
          . . . . . . . . . . . . . . . . . . . . . . . . . . .
        .                                                         .
        .                    .-(::::::::)                         .
        .                 .-(:::::::::::::)-.    +-+               .
        .                (:::: Segment A :::)--|R|---+            .
        .                 `-(:::::::::::::)-'    +-+   |           .
        .                    `-(::::::)-'               |          .
        .                                               |          .
        .                    .-(::::::::)               |          .
        .                 .-(:::::::::::::)-.    +-+     |          .
        .                (:::: Segment B :::)--|R|---+   |          .
        .                 `-(:::::::::::::)-'    +-+   |            .
        .                    `-(::::::)-'               |          .
        .                                               |          .
        .                    .-(::::::::)               |          .
        .                 .-(:::::::::::::)-.    +-+     |          .
        .                (:::: Segment C :::)--|R|---+   |          .
        .                 `-(:::::::::::::)-'    +-+   |            .
        .                    `-(::::::)-'               |          .
        .                                               |          .
        .                      ..(etc)..                X          .
        .                                                         .
        .                                                         .
        .          <- AERO Link Bridged by the SPAN ->            .
          . . . . . . . . . . . . . . .. . . . . . .
```
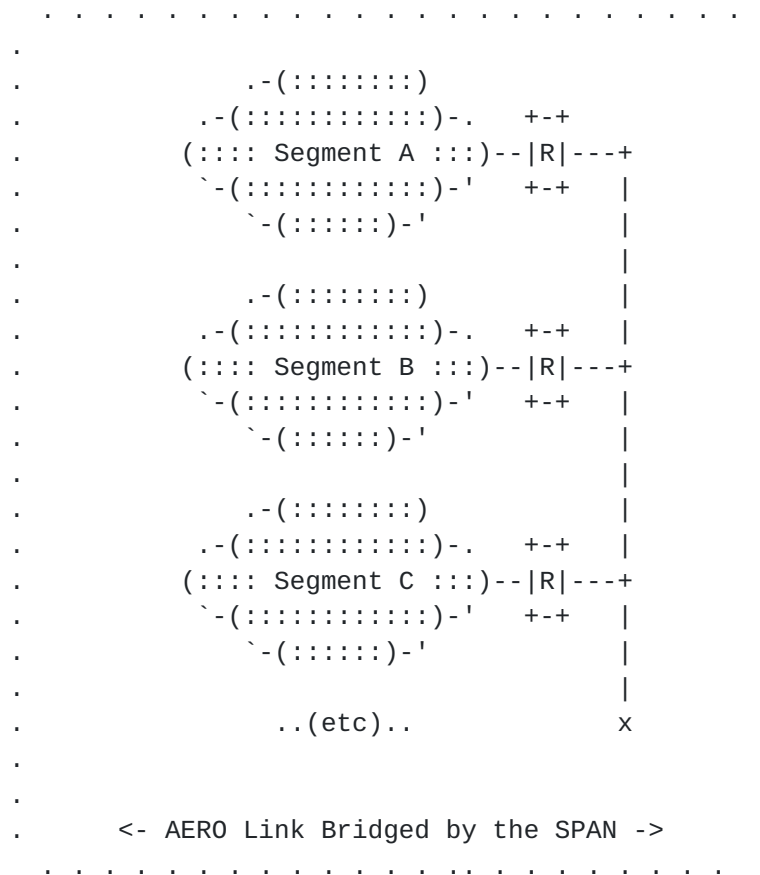
                        Figure 2: The SPAN

   To support the SPAN, AERO links require a reserved /96 IPv6 "SPAN
   Service Prefix (SSP)". Although any routable IPv6 prefix can be
   used, a Unique Local Address (ULA) prefix (e.g., fd00::/96) [RFC4389]
   is recommended since border routers are commonly configured to
   prevent packets with ULAs from being injected into the AERO link by
   an external IPv6 node and from leaking out of the AERO link to the
   outside world.

   Each segment in the SPAN assigns a unique sub-prefix of the SSP
   termed a "SPAN Partition Prefix (SPP)". For example, a first segment
   could assign fd00::1000/116, a second could assign fd00::2000/116, a
   third could assign fd00::3000/116, etc. The administrative
   authorities for each segment must therefore coordinate to assure
   mutually-exclusive SPP assignments, but internal provisioning of the
   SPP is a local consideration for each administrative authority.

   A "SPAN address" is an address taken from a SPP and assigned to a
   Relay, Server or Proxy interface. SPAN addresses are formed by
   simply replacing the upper portion of an administratively-assigned
   AERO address with the SPP. For example, if the SPP is

fd00::1000/116, the SPAN address formed from the AERO address
fe80::1001 is simply fd00::1001.

An "INET address" is an address of a node's interface connection to
an INET. AERO/SPAN/INET address mappings are maintained as permanent
neighbor cache entires as discussed in Section 3.8.

AERO Relays serve as bridges to join multiple segments into a unified
AERO link over multiple diverse administrative domains. They support
the bridging function by first establishing forwarding table entries
for their SPPs either via standard BGP routing or static routes. For
example, if three Relays ('A', 'B' and 'C') from different segments
serviced the SPPs fd00::1000/116, fd00::2000/116 and fd00::3000/116
respectively, then the forwarding tables in each Relay are as
follows:

A: fd00::1000/116->local, fd00::2000/116->B, fd00::3000/116->C

B: fd00::1000/116->A, fd00::2000/116->local, fd00::3000/116->C

C: fd00::1000/116->A, fd00::2000/116->B, fd00::3000/116->local

These forwarding table entries are permanent and never change, since
they correspond to fixed infrastructure elements in their respective
segments. This provides the basis for a link-layer forwarding
service that cannot be disrupted by routing updates due to node
mobility.

With the SPPs in place in each Relay's forwarding table, control and
data packets sent between AERO nodes in different segments can
therefore be carried over the SPAN via encapsulation. For example,
when a source node in segment A forwards a packet with IPv6 address
2001:db8:1:2::1 to a destination node in segment C with IPv6 address
2001:db8:1000:2000::1, it first encapsulates the packet in a SPAN
header with source SPAN address taken from fd00::1000/116 (e.g.,
fd00::1001) and destination SPAN address taken from fd00::3000/116
(e.g., fd00::3001). Next, it encapsulates the SPAN message in an
INET header with source address set to its own INET address (e.g.,
192.0.2.100) and destination set to the INET address of a Relay
(e.g., 192.0.2.1).

SPAN encapsulation is based on Generic Packet Tunneling in IPv6
[RFC2473]; the encapsulation format in the above example is shown
inFigure 3:

```
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |            INET Header        |
       |        src = 192.0.2.100      |
       |         dst = 192.0.2.1       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |            SPAN Header        |
       |        src = fd00::1001       |
       |        dst = fd00::3001       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |           Inner IP Header     |
       |     src = 2001:db8:1:2::1     |
       |   dst = 2001:db8:1000:2000::1 |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                               |
       ~                               ~
       ~          Inner Packet Body    ~
       ~                               ~
       |                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                   Figure 3: SPAN Encapsulation

   In this format, the inner IP header and packet body are the original
   IP packet, the SPAN header is an IPv6 header prepared according to
   [RFC2473], and the INET header is prepared according to Section 3.9.
   A packet is said to be "forwarded/sent into the SPAN" when it is
   encapsulated as described above then forwarded via a secured tunnel
   to a neighboring Relay.

   This gives rise to a routing system that contains both MNP routes
   that may change dynamically due to regional node mobility and SPAN
   routes that never change. The Relays can therefore provide link-
   layer bridging by sending packets into the SPAN instead of network-
   layer routing according to MNP routes. As a result, opportunities
   for packet loss due to node mobility between different segments are
   mitigated.

   With reference to Figure 3, for a Client's AERO address the SPAN
   address is simply set to the Subnet Router Anycast address. For non-
   link-local addresses, the destination SPAN address may not be known
   in advance for the first few packets of a flow sent via the SPAN. In
   that case, the SPAN destination address is set to the original
   packet's destination, and the SPAN routing system will direct the
   packet to the correct SPAN egress node. (In the above example, the
   SPAN destination address is simply 2001:db8:1000:2000::1.)

3.6. AERO Interface Characteristics

AERO interfaces use encapsulation (see: Section 3.9) to exchange
packets with neighbors attached to the AERO link.

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

AERO interface ND messages include one or more Source/Target Link-
Layer Address Options (S/TLLAOs) formatted as shown in Figure 4:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |  Length = 5   | Prefix Length |S|R|D|X|N|Resvd|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Interface ID          |         Port Number           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   |                                                               |
   +                     Link Layer Address                        +
   |                                                               |
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |P00|P01|P02|P03|P04|P05|P06|P07|P08|P09|P10|P11|P12|P13|P14|P15|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |P16|P17|P18|P19|P20|P21|P22|P23|P24|P25|P26|P27|P28|P29|P30|P31|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |P32|P33|P34|P35|P36|P37|P38|P39|P40|P41|P42|P43|P44|P45|P46|P47|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |P48|P49|P50|P51|P52|P53|P54|P55|P56|P57|P58|P59|P60|P61|P62|P63|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
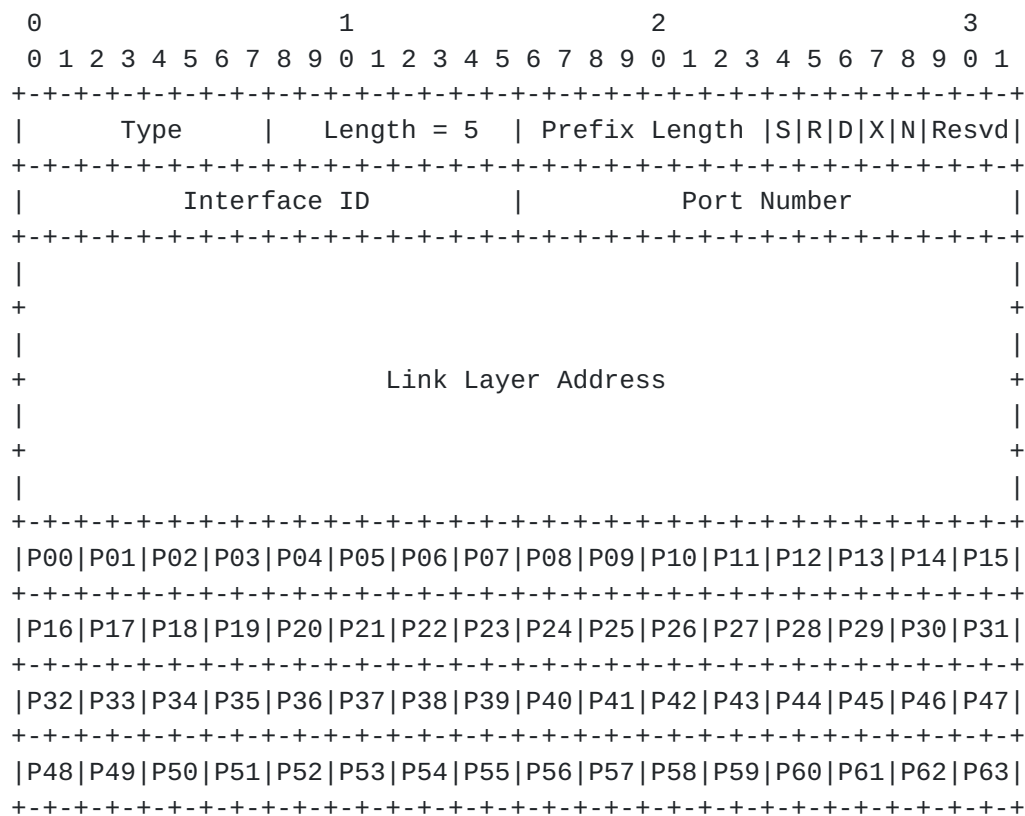
Figure 4: AERO Source/Target Link-Layer Address Option (S/TLLAO)
                           Format

In this format:

o Type is set to '1' for SLLAO or '2' for TLLAO.

o Length is set to the constant value '5' (i.e., 5 units of 8
   octets).

o Prefix Length is set to the MNP prefix length in an ND message for
   the Client AERO address found in the source (RS), destination (RA)
   or target (NA) address; otherwise set to 0. If the message
   contains multiple SLLAOs, only the Prefix Length value in the
   SLLAO with S set to 1 is consulted and the values in other SLLAOs
   are ignored.

o S (the 'Source' bit) is set to '1' in the S/TLLAO of an ND message
   that corresponds to the ANET/INET interface over which the ND
   message is sent, and set to 0 in all other S/TLLAOs.

o R (the "Release" bit) is set to '1' in an S/TLLAO in an RS/NA sent
   for the purpose of departing from a Server; otherwise, set to '0'.
   The recipient places the corresponding neighbor cache entry in the
   DEPARTED state. For RS message, the recipient then releases the
   corresponding PD and returns an RA with Router Lifetime set to '0'

o D (the "Disable" bit) is set to '1' in the S/TLLAOs of an RS/NA
   message for each Interface ID that is to be disabled in the
   neighbor cache entry; otherwise, set to '0'.

o X (the "proXy" bit) is set to '1' in the SLLAO of an RS/RA message
   by the Proxy when there is a Proxy in the path; otherwise, set to
   '0'. If the message contains multiple SLLAOs, only the X value in
   the SLLAO with S set to 1 is consulted and the values in other
   SLLAOs are ignored.

o N (the "(Network Address) Translator (NAT)" bit) is set to '1' in
   the SLLAO of an RA message by the Server if there is a translator
   in the path; otherwise, set to '0'. If the message contains
   multiple SLLAOs, only the N value in the SLLAO with S set to 1 is
   consulted and the values in other SLLAOs are ignored.

o Resvd is set to the value '0' on transmission and ignored on
   receipt.

o Interface ID is set to a 16-bit integer value corresponding to an
   AERO node's ANET/INET interface. Once the node has assigned an
   Interface ID to an ANET interface, the assignment must remain
   unchanged until the node fully detaches from the AERO link. The
   value 0xffff is reserved as the Server's INET Interface ID, i.e.,
   Servers MUST use Interface ID 0xffff, and Clients MUST number
   their ANET Interface IDs with values in the range of 0-0xfffe.

o Port Number and Link Layer Address are set to the encapsulation
   addresses required to send packets via the target node (or to '0'
   when the addresses are left unspecified). When UDP is not used as
   part of the encapsulation, Port Number is set to '0'. When the

encapsulation IP address family is IPv4, IP Address is formed as
an IPv4-mapped IPv6 address as specified in Section 3.4.

o P(i) is a set of Preferences that correspond to the 64
  Differentiated Service Code Point (DSCP) values [RFC2474]. Each
  P(i) is set to the value '0' ("disabled"), '1' ("low"), '2'
  ("medium") or '3' ("high") to indicate a QoS preference level for
  packet forwarding purposes.

A Client's AERO interface may be configured over multiple ANET
interface connections. For example, common mobile handheld devices
have both wireless local area network ("WLAN") and cellular wireless
links. These links are typically used "one at a time" with low-cost
WLAN preferred and highly-available cellular wireless as a standby.
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.

A Client's ANET interfaces are classified as follows:

o Native interfaces connect to the open INET, and have a global IP
  address that is reachable from any INET correspondent.

o NATed interfaces connect to an ANET behind a Network Address
  Translator (NAT). The NAT does not participate in any AERO
  control message signaling, but the Server can issue control
  messages on behalf of the Client. Clients that are behind a NAT
  are required to send periodic keepalive messages to keep NAT state
  alive when there are no data packets flowing. If no other
  periodic messaging service is available, the Client can send RS
  messages to receive RA replies from its Server(s).

o VPNed interfaces use security encapsulation over the ANET to a
  Virtual Private Network (VPN) server that also acts as an AERO
  Server. As with NATed links, the Server can issue control
  messages on behalf of the Client, but the Client need not send
  periodic keepalives in addition to those already used to maintain
  the VPN connection.

o Proxyed interfaces connect to an ANET that is separated from the
  open INET by an AERO Proxy. Unlike NATed and VPNed interfaces,
  the Proxy can actively issue control messages on behalf of the
  Client.

o Direct interfaces connect the Client directly to a neighbor
  without crossing any ANET/INET paths. An example is a line-of-
  sight link between a remote pilot and an unmanned aircraft.

If a Client's multiple ANET interfaces are used "one at a time" (i.e., all other interfaces are in standby mode while one interface is active), then ND messages include only a single S/TLLAO with Interface ID set to a constant value. In that case, the Client would appear to have a single ANET interface but with a dynamically changing ANET address.

If the Client has multiple active ANET interfaces, then from the perspective of ND it would appear to have multiple link-layer addresses. In that case, ND messages MAY include multiple S/TLLAOs -- each with an Interface ID that corresponds to a specific ANET interface. S must be set to 1 in the S/TLLAO corresponding to the AERO node's ANET interface used to transmit the message and set to 0 in all other S/TLLAOs.

When the Client includes an S/TLLAO for an ANET interface for which it is aware that there is a NAT on the path to the Server, or when a node includes an S/TLLAO solely for the purpose of announcing new QoS preferences, the node MAY set both Port Number and Link-Layer Address to 0 to indicate that the addresses are unspecified at the network layer and must instead be derived from the link-layer encapsulation headers.

Relay, Server and Proxy AERO interfaces may be configured over one or more secured tunnel interfaces. The AERO interface configures both an AERO address and its corresponding SPAN address, while the underlying secured tunnel interfaces are either unnumbered or configure the same SPAN address. The AERO interface encapsulates each IP packet in a SPAN header and presents the packet to the underlying secured tunnel interface. For Relays that do not configure an AERO interface, the secured tunnel interfaces themselves are exposed to the IP layer with each interface configuring the Relay's SPAN address. Routing protocols such as BGP therefore run directly over the Relay's secured tunnel interfaces. For nodes that configure an AERO interface, routing protocols such as BGP run over the AERO interface but do not employ SPAN encapsulation. Instead, the AERO interface presents the routing protocol messages directly to the underlying secured tunnels without applying encapsulation and while using the SPAN address 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.

3.7. AERO Interface Initialization

AERO Servers, Proxys and Clients configure AERO interfaces as their point of attachment to the AERO link. AERO nodes assign the MSPs for the link to their AERO interfaces (i.e., as a "route-to-interface")

to ensure that packets with destination addresses covered by an MNP
not explicitly assigned to a non-AERO interface are directed to the
AERO interface.

AERO interface initialization procedures for Servers, Proxys, Clients
and Relays are discussed in the following sections.

3.7.1. AERO Server/Gateway Behavior

When a Server enables an AERO interface, it assigns AERO/SPAN
addresses and configures permanent neighbor cache entries for
neighbors in the same SPAN segment by consulting the ROS list for the
segment. The Server also configures secured tunnels with one or more
neighboring Relays and engages in a BGP routing protocol session with
each Relay.

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

Gateways are simply Servers that run a dynamic routing protocol
between the AERO interface and INET/EUN interfaces (see:
Section 3.3). The Gateway 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 AERO link over the INET/EUN interfaces. The
Gateway further provides an attachment point of the AERO link to the
non-MNP-based global topology.

3.7.2. AERO Proxy Behavior

When a Proxy enables an AERO interface, it assigns AERO/SPAN
addresses and configures permanent neighbor cache entries the same as
for Servers. The Proxy also configues secured tunnels with one or
more neighboring Relays and maintains per-Client neighbor cache
entries based on control message exchanges.

3.7.3. AERO Client Behavior

When a Client enables an AERO interface, it sends RS messages with
ND/PD parameters over an ANET interface to one or more Servers in the
MAP list, which return RA messages with corresponding PD parameters.
(The RS/RA messages may pass through a Proxy in the case of a
Client's Proxyed interface.)

After the initial ND/PD message exchange, the Client assigns AERO
addresses to the AERO interface based on the delegated prefix(es).
The Client can then register additional ANET interfaces with the
Server by sending an RS message over each ANET interface.

3.7.4. AERO Relay Behavior

AERO Relays need not connect directly to the AERO link, since they
operate as link-layer forwarding devices instead of network layer
routers. Configuration of AERO interfaces on Relays is therefore
OPTIONAL, e.g., if an administrative interface is needed. Relays
configure secured tunnels with Servers, Proxys and other Relays; they
also configure AERO/SPAN addresses and permanent neighbor cache
entries the same as Servers. Relays engage in a BGP routing protocol
session with a subset of the Servers on the local SPAN segment, and
with other Relays on the SPAN (see: Section 3.3).

3.8. AERO Interface Neighbor Cache Maintenance

Each AERO interface maintains a conceptual neighbor cache that
includes an entry for each neighbor it communicates with on the AERO
link per [RFC4861]. AERO interface neighbor cache entries are said
to be one of "permanent", "symmetric", "asymmetric" or "proxy".

Permanent neighbor cache entries are created through explicit
administrative action; they have no timeout values and remain in
place until explicitly deleted. AERO Servers and Proxys maintain
permanent neighbor cache entries for all other Servers and Proxys
within the same SPAN segment. Each entry maintains the mapping
between the neighbor's network-layer AERO address and corresponding
INET address. The list of all permanent neighbor cache entries for
the SPAN segment is maintained in the segment's ROS list.

Symmetric neighbor cache entries are created and maintained through
RS/RA exchanges as specified in Section 3.15, and remain in place for
durations bounded by ND/PD lifetimes. AERO Servers maintain
symmetric neighbor cache entries for each of their associated
Clients, and AERO Clients maintain symmetric neighbor cache entries
for each of their associated Servers. The list of all Servers on the
AERO link is maintained in the link's MAP list.

Asymmetric neighbor cache entries are created or updated based on
route optimization messaging as specified in Section 3.17, and are
garbage-collected when keepalive timers expire. AERO route
optimization sources (ROSs) maintain asymmetric neighbor cache
entries for active targets with lifetimes based on ND messaging
constants. Asymmetric neighbor cache entries are unidirectional

since only the ROS and not the target (e.g., a Client's MAP) creates
an entry.

Proxy neighbor cache entries are created and maintained by AERO
Proxys when they process Client/Server ND/PD exchanges, and remain in
place for durations bounded by ND/PD lifetimes. AERO Proxys maintain
proxy neighbor cache entries for each of their associated Clients.
Proxy neighbor cache entries track the Client state and the address
of the Client's associated Server.

To the list of neighbor cache entry states in Section 7.3.2 of
[RFC4861], AERO interfaces add an additional state DEPARTED that
applies to symmetric and proxy neighbor cache entries for Clients
that have recently departed. The interface sets a "DepartTime"
variable for the neighbor cache entry to "DEPARTTIME" 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, 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 DEPARTTIME be set to the default constant value 40
seconds to allow for packets in flight to be delivered while stale
route optimization state may be present.

When a target Server (acting as a Mobility Anchor Point (MAP))
receives a valid NS message used for route optimization, it searches
for a symmetric neighbor cache entry for the target Client. The MAP
then returns a solicited NA message without creating a neighbor cache
entry for the ROS, but creates a target Client "Report List" entry
for the ROS and sets a "ReportTime" variable for the entry to
REPORTTIME seconds. The MAP resets ReportTime when it receives a new
authentic NS message, and otherwise decrements ReportTime while no NS
messages have been received. It is RECOMMENDED that REPORTTIME be
set to the default constant value 40 seconds to allow a 10 second
window so that route optimization can converge before ReportTime
decrements below REACHABLETIME.

When the ROS receives a solicited NA message response to its NS
message, it creates or updates an asymmetric neighbor cache entry for
the target network-layer and link-layer addresses. The ROS then
(re)sets ReachableTime for the neighbor cache entry to REACHABLETIME
seconds and uses this value to determine whether 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 REACHABLETIME
be set to the default constant value 30 seconds as specified in
[RFC4861].

The ROS also uses 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 DEPARTTIME, REPORTTIME, REACHABLETIME,
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, DEPARTTIME and
REPORTTIME SHOULD be set to a value that is sufficiently longer than
REACHABLETIME to avoid packet loss due to stale route optimization
state.

3.9. AERO Interface Encapsulation and Re-encapsulation

AERO interfaces encapsulate packets according to whether they are
entering the AERO interface from the network layer or if they are
being re-admitted into the same AERO link they arrived on. This
latter form of encapsulation is known as "re-encapsulation".

For packets entering the AERO interface from the network layer, the
AERO interface copies the "TTL/Hop Limit", "Type of Service/Traffic
Class" [RFC2983], "Flow Label"[RFC6438] (for IPv6) and "Congestion
Experienced" [RFC3168] values in the packet's IP header into the
corresponding fields in the encapsulation header(s).

For packets undergoing re-encapsulation, the AERO interface instead
copies these values from the original encapsulation header into the
new encapsulation header, i.e., the values are transferred between
encapsulation headers and *not* copied from the encapsulated packet's
network-layer header. (Note especially that by copying the TTL/Hop
Limit between encapsulation headers the value will eventually
decrement to 0 if there is a (temporary) routing loop.) For IPv4
encapsulation/re-encapsulation, the AERO interface sets the DF bit as
discussed in Section 3.13.

Proxy and Relay AERO interfaces encapsulate each packet in a SPAN
header, then encapsulate the resulting SPAN packet in an INET header
according to the next hop determined in the forwarding algorithm in
Section 3.12. If the next hop is reached via a secured tunnel, the
AERO interface uses an INET encapsulation format specific to the
secured tunnel type (see: Section 6). If the next hop is reached via
an unsecured underlying interface, the AERO interface instead uses

Generic UDP Encapsulation (GUE) [I-D.ietf-intarea-gue] or an
alternate minimal encapsulation format Appendix A.

Client AERO interfaces can avoid encapsulation over underlying
interfaces for which the first-hop access router is AERO-aware. For
other underlying interfaces, Client interfaces use INET encapsulation
the same as for Relays/Proxys but do not use SPAN encapsulation
unless the underlying interface itself connects to the SPAN.

When GUE encapsulation is used, the AERO interface next sets the UDP
source port to a constant value that it will use in each successive
packet it sends, and sets the UDP length field to the length of the
SPAN packet plus 8 bytes for the UDP header itself plus the length of
the GUE header (or 0 if GUE direct IP encapsulation is used). For
packets sent to a Server or Relay, the AERO interface sets the UDP
destination port to 8060, i.e., the IANA-registered port number for
AERO. For packets sent to a Client, the AERO interface sets the UDP
destination port to the port value stored in the neighbor cache entry
for this Client. The AERO interface then either includes or omits
the UDP checksum according to the GUE specification.

AERO interfaces observes the packet sizing and fragmentation
considerations found in Section 3.13.

3.10. AERO Interface Decapsulation

AERO interfaces decapsulate packets destined either to the AERO node
itself or to a destination reached via an interface other than the
AERO interface the packet was received on. When the encapsulated
packet arrives in multiple fragments, the AERO interface reassembles
as discussed in Section 3.13. Further decapsulation steps are
performed according to the appropriate encapsulation format
specification.

3.11. AERO Interface Data Origin Authentication

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

o AERO Relays, Servers and Proxys accept encapsulated data packets
   and control messages received from secured tunnels.

o AERO Servers and Proxys accept encapsulated data packets and NS
   messages used for Neighbor Unreachability Detection (NUD) received
   from a source found in the ROS list.

o AERO Proxys and Clients accept packets that originate from within
   the same secured ANET.

   o AERO Clients and Gateways accept packets from downstream network
      correspondents based on ingress filtering.

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

3.12. AERO Interface Forwarding Algorithm

   IP packets enter a node's AERO interface either from the network
   layer (i.e., from a local application or the IP forwarding system) or
   from the link layer (i.e., from an AERO interface neighbor). All
   packets entering a node's AERO interface first undergo data origin
   authentication as discussed in Section 3.11. Packets that satisfy
   data origin authentication are processed further, while all others
   are dropped silently.

   Packets that enter the AERO interface from the network layer are
   forwarded to an AERO interface neighbor. Packets that enter the AERO
   interface from the link layer are either re-admitted into the AERO
   link or forwarded to the network layer where they are subject to
   either local delivery or IP forwarding. In all cases, the AERO
   interface itself MUST NOT decrement the network layer TTL/Hop-count
   since its forwarding actions occur below the network layer.

   AERO interfaces may have multiple underlying ANET/INET interfaces
   and/or neighbor cache entries for neighbors with multiple Interface
   ID registrations (see Section 3.6). The AERO interface uses each
   packet's DSCP value (and/or port number) to select an outgoing ANET/
   INET interface based on the node's own QoS preferences, and also to
   select a destination link-layer address based on the neighbor's ANET/
   INET 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 packet and sends
   one copy via each of the (outgoing / neighbor) interface pairs;
   otherwise, the node sends a single copy of the packet via the
   interface with the highest preference. AERO nodes keep track of
   which ANET/INET interfaces are currently "reachable" or
   "unreachable", and only use "reachable" interfaces for forwarding
   purposes.

   The following sections discuss the AERO interface forwarding
   algorithms for Clients, Proxys, Servers and Relays. In the following
   discussion, a 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 AERO address).

3.12.1. Client Forwarding Algorithm

When an IP packet enters a Client's AERO interface from the network
layer the Client searches for an asymmetric neighbor cache entry that
matches the destination. If there is a match, the Client uses one or
more "reachable" neighbor interfaces in the entry for packet
forwarding. If there is no asymmetric neighbor cache entry, the
Client instead forwards the packet toward a Server (the packet is
intercepted by a Proxy if there is a Proxy on the path).

When an IP packet enters a Client's AERO interface from the link-
layer, if the destination matches one of the Client's MNPs or link-
local addresses the Client decapsulates the packet (if necessary) and
delivers it to the network layer. Otherwise, the Client drops the
packet and MAY return a network-layer ICMP Destination Unreachable
message subject to rate limiting (see: Section 3.14).

3.12.2. Proxy Forwarding Algorithm

For control messages originating from or destined to a Client, the
Proxy intercepts the message and updates its proxy neighbor cache
entry for the Client. The Proxy then forwards a (proxyed) copy of
the control message. (For example, the Proxy forwards a proxied
version of a Client's NS/RS message to the target neighbor, and
forwards a proxied version of the NA/RA reply to the Client.)

When the Proxy receives a data packet from a Client within the ANET,
the Proxy searches for an asymmetric neighbor cache entry that
matches the destination and forwards the packet as follows:

o if the destination matches an asymmetric neighbor cache entry, the
   Proxy uses one or more "reachable" neighbor interfaces in the
   entry for packet forwarding via encapsulation. If the neighbor
   interface is in the same SPAN segment, the Proxy forwards the
   packet directly to the neighbor; otherwise, it forwards the packet
   to a Relay.

o else, the Proxy encapsulates and forwards the packet to a Relay
   while using the packet's destination address as the SPAN
   destination address. (If the destination is an AERO address, the
   Proxy instead uses the corresponding Subnet Router Anycast address
   for Client AERO addresses and the SPAN address for
   administratively-provisioned AERO addresses.).

When the Proxy receives an encapsulated data packet from an INET
neighbor or from a secured tunnel, it accepts the packet only if data
origin authentication succeeds and the SPAN destination address is
its own adddress. If the packet is a SPAN fragment, the Proxy then
adds the fragment to the reassembly buffer and returns if the
reassembly is still incomplete. Otherwise, the Proxy reassembles the
packet (if necessary) and continues processing.

Next, the Proxy searches for a proxy neighbor cache entry that
matches the destination. If there is a proxy neighbor cache entry in
the REACHABLE state, the Proxy decapsulates and forwards the packet
to the Client. If the neighbor cache entry is in the DEPARTED state,
the Proxy instead re-encpasulates the message with the address of the
Client's Server as the SPAN destination address and forwards the
packet to a Relay. The Proxy then returns an unsolicited NA message
as discussed in Section 3.19. If there is no neighbor cache entry,
the Proxy instead discards the packet.

3.12.3. Server/Gateway Forwarding Algorithm

For control messages destined to a target Client's AERO address that
are received from a secured tunnel, the Server (acting as a MAP)
intercepts the message and sends an appropriate response on behalf of
the Client. (For example, the Server sends an NA message reply in
response to an NS message directed to one of its associated Clients.)
If the Client's neighbor cache entry is in the DEPARTED state,
however, the Server instead forwards the packet to the Client's new
Server as discussed in Section 3.19.

When the Server receives an encapsulated data packet from an INET
neighbor or from a secured tunnel, it accepts the packet only if data
origin authentication succeeds. If the SPAN destination address is
its own adddress, the Server reassembles if necessary and discards
the SPAN header (if the reassembly is incomplete, the Server instead
adds the fragment to the reassembly buffer and returns). The Server
then continues processing as follows:

o if the destination matches a symmetric neighbor cache entry in the
   REACHABLE state the Server prepares the packet for forwarding to
   the destination Client. If the current header is a SPAN header,
   the Server reassembles if necessary and discards the SPAN header
   (if the reassembly is incomplete, the Server instead adds the
   fragment to the reassembly buffer and returns). The Server then
   forwards the packet according to the cached link-layer
   information, while using SPAN encapsulation for the Client's
   Proxyed/Native interfaces, simple INET encapsulation for NATed/
   VPNed interfaces, or no encapsulation for Direct interfaces. If
   the packet is destined to the same Client from which it arrived

(i.e., if the packet was forwarded by one of the Client's Proxys),
the Server forwards the packet via a different "reachable"
neighbor interface than the one the packet arrived on. If there
are no "reachable" neighbor interfaces, the Server drops the
packet.

o else, if the destination matches a symmetric neighbor cache entry
   in the DEPARETED state the Server encapsulates the packet in a new
   SPAN header and forwards it to the Client's new Server (noting
   that the encapsulation may result in the addition of a second SPAN
   header). The Server uses its own SPAN address as the source and
   the SPAN address of the new Server as the destination.

o else, if the destination matches an asymmetric neighbor cache
   entry, the Server uses one or more "reachable" neighbor interfaces
   in the entry for packet forwarding via the local INET if the
   neighbor is in the same SPAN segment or via a Relay otherwise.

o else, if the destination is an AERO address that is not assigned
   on the AERO interface the Server drops the packet.

o else, the Server (acting as a Gateway) releases the packet to the
   network layer for local delivery or IP forwarding. Based on the
   information in the forwarding table, the network layer may return
   the packet to the same AERO interface in which case further
   processing occurs as below. (Note that this arrangement
   accommodates common implementations in which the IP forwarding
   table is not accessible from within the AERO interface. If the
   AERO interface can directly access the IP forwarding table, the
   forwarding table lookup can instead be performed internally from
   within the AERO interface itself.)

When the Server's AERO interface receives a data packet from the
network layer or from a NATed/VPNed/Direct Client, it processes the
packet according to the network-layer destination address as follows:

o if the destination matches a symmetric or asymmetric neighbor
   cache entry the Server processes the packet as above.

o else, the Server encapsulates the packet and forwards it to a
   Relay. For administratively-assigned AERO address destinations,
   the Server uses the SPAN address corresponding to the destination
   as the SPAN destination address. For Client AERO address
   destinations, the Server uses the Subnet Router Anycast address
   corresponding to the destination as the SPAN destination address.
   For all others, the Server uses the packet's destination IP
   address as the SPAN destination address.

[3.12.4](#). Relay Forwarding Algorithm

Relays forward packets over secured tunnels the same as any IP
router. When the Relay receives an encapsulated packet via a secured
tunnel, it removes the INET header and searches for a forwarding
table entry that matches the destination address in the next header.
The Relay then processes the packet as follows:

o if the destination matches one of the Relay's own addresses, the
   Relay submits the packet for local delivery.

o else, if the destination matches a forwarding table entry the
   Relay forwards the packet via a secured tunnel to the next hop.
   If the destination matches an MSP without matching an MNP,
   however, the Relay instead drops the packet and returns an ICMP
   Destination Unreachable message subject to rate limiting (see:
   [Section 3.14](#)).

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

As for any IP router, the Relay decrements the TTL/Hop Limit when it
forwards the packet. If the packet is encapsulated in a SPAN header,
only the Hop Limit in the SPAN header is decremented, and not the
TTL/Hop Limit in the inner packet header.

[3.13](#). AERO Interface MTU and Fragmentation

The AERO interface is the node's attachment to the AERO link. For
AERO link neighbor underlying interface paths that do not require
encapsulation, the AERO interface sends unencapsulated IP packets.
For other paths, the AERO interface acts as a tunnel ingress when it
sends packets to the neighbor and as a tunnel egress when it receives
packets from the neighbor. AERO interfaces observe the packet sizing
considerations for tunnels discussed in [[I-D.ietf-intarea-tunnels](#)]
and as specified below.

The Internet Protocol expects that IP packets will either be
delivered to the destination or a suitable Packet Too Big (PTB)
message returned to support the process known as IP Path MTU
Discovery (PMTUD) [[RFC1191](#)][RFC8201]. However, PTB messages may be
crafted for malicious purposes or lost in the network [[RFC2923](#)].
This can be especially problematic for tunnels, where a condition
known as a PMTUD "black hole" can result. For these reasons, AERO
interfaces employ operational procedures that avoid interactions with
PMTUD, including the use of fragmentation when necessary.

AERO interfaces observe two different types of fragmentation. Source
fragmentation occurs when the AERO interface (acting as a tunnel
ingress) fragments the encapsulated packet into multiple fragments
before admitting each fragment into the tunnel. Network
fragmentation occurs when an encapsulated packet admitted into the
tunnel by the ingress is fragmented by an IPv4 router on the path to
the egress. Note that an IPv4 packet that incurs source
fragmentation may also incur network fragmentation.

IPv6 specifies a minimum link Maximum Transmission Unit (MTU) of 1280
bytes [RFC8200]. Although IPv4 specifies a smaller minimum link MTU
of 68 bytes [RFC0791], AERO interfaces also observe the IPv6 minimum
for IPv4 even if encapsulated packets may incur network
fragmentation.

IPv6 specifies a minimum Maximum Reassembly Unit (MRU) of 1500 bytes
[RFC8200], while the minimum MRU for IPv4 is only 576 bytes [RFC1122]
(but, note that many standard IPv6 over IPv4 tunnel types already
assume a larger MRU than the IPv4 minimum).

AERO interfaces therefore configure an MTU that MUST NOT be smaller
than 1280 bytes, MUST NOT be larger than the minimum MRU among all
nodes on the AERO link minus the encapsulation overhead ("ENCAPS"),
and SHOULD NOT be smaller than 1500 bytes. AERO interfaces also
configure a Maximum Segment Unit (MSU) as the maximum-sized
encapsulated packet that the ingress can inject into the tunnel
without source fragmentation. The MSU value MUST NOT be larger than
(MTU+ENCAPS) and MUST NOT be larger than 1280 bytes unless there is
operational assurance that a larger size can traverse the link along
all paths. The MSU value SHOULD also be reduced by 40 bytes to
accommodate the possibility that an additional encapsulation is added
(e.g., see: Section 3.12.3).

All AERO nodes MUST configure the same MTU value for reasons cited in
[RFC3819][RFC4861]; in particular, multicast support requires a
common MTU value among all nodes on the link. All AERO nodes MUST
configure an MRU large enough to reassemble packets up to
(MTU+ENCAPS) bytes in length; nodes that cannot configure a large-
enough MRU MUST NOT enable an AERO interface. For example, for an
MTU of 1500 bytes (or slightly larger) an appropriate MRU might be
2KB.

The network layer proceeds as follows when it forwards an IP packet
to the AERO interface. For each IPv4 packet that is larger than the
AERO interface MTU and with DF set to 0, the network layer uses IPv4
fragmentation to break the packet into a minimum number of non-
overlapping fragments where the first fragment is no larger than the
MTU and the remaining fragments are no larger than the first. For

all other IP packets, if the packet is larger than the AERO interface
MTU, the network layer drops the packet and returns a PTB message to
the original source. Otherwise, the network layer admits each IP
packet or fragment into the AERO interface.

For each IP packet admitted into the AERO interface, if the neighbor
is reached via an underlying interface that does not require
encapsulation the AERO interface proceeds according to the underlying
interface MTU. If the packet is no larger than the underlying
interface MTU, the AERO interface presents the packet to the
underlying interface. Otherwise, for IPv4 packets with DF set to 0
the AERO interface uses IPv4 fragmentation to break the packet into
fragments no larger than the underlying interface MTU. For other
packets, the AERO interface drops the packet and returns a PTB
message to the original source. (If the original source corresponds
to a local application, the PTB would appear to have originated from
a router on the path when in fact it was locally generated from
within the AERO interface.)

For underlying interfaces that require encapsulation, the AERO
interface (acting as a tunnel ingress) instead encapsulates the
packet in a SPAN heaedr. If the SPAN packet is larger than the MSU,
the ingress source fragments the SPAN packet into a minimum number of
non-overlapping and (roughly) equal-length fragments where the first
fragment is no larger than the MSU and the remaining fragments are no
larger than the first. The ingress then encapsulates each SPAN
packet/fragment in an INET header and admits them into the tunnel.
For IPv4, the ingress sets the DF bit to 0 in the INET header in case
any network fragmentation is necessary. The encapsulated packets
will be delivered to the egress, which reassembles them into a whole
packet if necessary.

By fragmenting at the SPAN layer instead of lower layers, standard
IPv6 fragmentation and reassembly [RFC8200] ensures that IPv4 issues
such as data corruption due to reassembly misassociations will not
occur [RFC6864][RFC4963]. Incomplete reassemblies are discarded when
the corresponding neighbor cache entry is deleted.

3.14. AERO Interface Error Handling

When an AERO node admits encapsulated packets into the AERO
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]. (AERO interfaces ignore all link-layer IPv4
"Fragmentation Needed" and IPv6 "Packet Too Big" messages since they
only emit packets that are guaranteed to be no larger than the IP
minimum link MTU as discussed in Section 3.13.)

The ICMP header is followed by the leading portion of the 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"
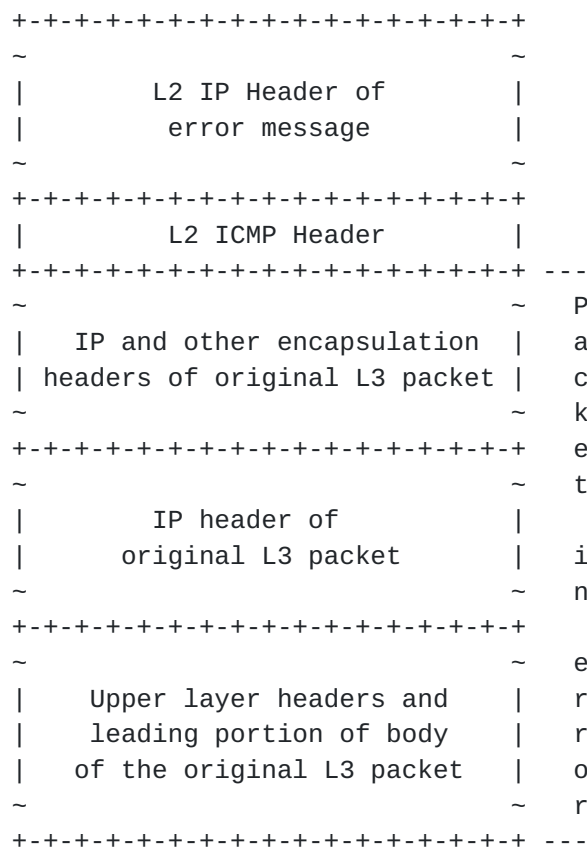and "L3" refer to link-layer and network-layer, respectively):

```
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     ~                               ~
     |        L2 IP Header of        |
     |         error message         |
     ~                               ~
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |         L2 ICMP Header        |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
     ~                               ~  P
     |   IP and other encapsulation  |  a
     | headers of original L3 packet |  c
     ~                               ~  k
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  e
     ~                               ~  t
     |         IP header of          |
     |       original L3 packet      |  i
     ~                               ~  n
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     ~                               ~  e
     |      Upper layer headers and  |  r
     |      leading portion of body  |  r
     |     of the original L3 packet |  o
     ~                               ~  r
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
```

          Figure 5: AERO 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 encapsulated packets that it
      sends to one of its asymmetric neighbor correspondents, the node
      SHOULD process the message as an indication that a path may be
      failing, and MAY initiate NUD over that path. If it receives
      Destination Unreachable messages on many or all paths, the node
      SHOULD set ReachableTime for the corresponding asymmetric neighbor

cache entry to 0 and allow future packets destined to the
correspondent to flow through a default route.

o When an AERO Client receives persistent link-layer Destination
   Unreachable messages in response to encapsulated packets that it
   sends to one of its symmetric neighbor 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 Server and release its association
   with the old Server as specified in Section 3.19.5.

o When an AERO Server receives persistent link-layer Destination
   Unreachable messages in response to encapsulated packets that it
   sends to one of its symmetric neighbor Clients, the Server SHOULD
   mark the underlying path as unusable and use another underlying
   path. If it receives Destination Unreachable messages on multiple
   paths, the Server should take no further actions unless it
   receives an explicit ND/PD release message or if the PD lifetime
   expires. In that case, the Server MUST release the Client's
   delegated MNP, withdraw the MNP from the AERO routing system and
   delete the neighbor cache entry.

o When an AERO Server or Proxy receives link-layer Destination
   Unreachable messages in response to an encapsulated 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 Relay receives a packet for which the network-layer
destination address is covered by an MSP, if there is no more-
specific routing information for the destination the Relay drops the
packet and returns a network-layer Destination Unreachable message
subject to rate limiting. The Relay writes the network-layer source
address of the original 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 an encapsulated 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 packet as the destination
address and writes one of its non link-local addresses as the source
address.

3.15. AERO Router Discovery, Prefix Delegation and Autoconfiguration

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

3.15.1. AERO ND/PD Service Model

Each AERO Server on the link configures a PD service to facilitate
Client requests. Each 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 AERO link and securely distributed to all Servers,
e.g., via the Lightweight Directory Access Protocol (LDAP) [RFC4511],
via static configuration, etc. Clients can receive new PDs from new
Servers before releasing PDs received from existing Servers for
service continuity. Clients receive the same service regardless of
the Servers they select, although selecting Servers that are
topologically nearby may provide better routing.

AERO Clients and Servers use ND messages to maintain neighbor cache
entries. AERO Servers configure their AERO interfaces as advertising
interfaces, and therefore send unicast RA messages with configuration
information in response to a Client's RS message. Thereafter,
Clients send additional RS messages to refresh prefix and/or router
lifetimes.

AERO Clients and Servers include PD parameters in RS/RA messages (see
[I-D.templin-6man-dhcpv6-ndopt] for ND/PD alternatives). The unified
ND/PD messages are exchanged between Client and Server according to
the prefix management schedule required by the PD service. If the
Client knows its MNP in advance, it can include its AERO address as
the source address of an RS message and with an SLLAO with a valid
Prefix Length for the MNP. If the Server (and Proxy) accept the
Client's MNP assertion, they inject the prefix into the routing
system and establish the necessary neighbor cache state.

The following sections specify the Client and Server behavior.

3.15.2. AERO Client Behavior

AERO Clients can discover the addresses of Servers in the MAP list
via static configuration (e.g., from a flat-file map of Server
addresses and locations), or through an automated means such as
Domain Name System (DNS) name resolution [RFC1035]. 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 AERO link (e.g., "example.com"). Alternatively,
the Client can discover the Server's address through a multicast RS
as described below.

To associate with a Server, the Client acts as a requesting router to
request MNPs. The Client prepares an RS message with PD parameters
(e.g., with an SLLAO with non-zero Prefix Length) and SHOULD include
a Nonce and Timestamp option if the Client needs to correlate RA
replies. If the Client already knows the Server's AERO address, it
includes the AERO address as the network-layer destination address;
otherwise, it includes all-routers multicast (ff02::2) as the
network-layer destination address. If the Client already knows its
own AERO address, it uses the AERO address as the network-layer
source address; otherwise, it uses the unspecified AERO address
(fe80::ffff:ffff) as the network-layer source address.

The Client next includes an SLLAO in the RS message formatted as
described in Section 3.6 to register its link-layer information with
the Server. The SLLAO corresponding to the ANET interface over which
the Client will send the RS message MUST set S to 1. The Client MAY
include additional SLLAOs specific to other underlying interfaces,
but if so it MUST set their S, Port Number and Link Layer Address
fields to 0. If the Client is connected to an ANET for which
encapsulation is required, the Client finally encapsulates the RS
message in an ANET header with its own ANET address as the source
address and the INET address of the Server as the destination.

The Client then sends the RS message (either directly via Direct
interfaces, via INET encapsulation for NATed interfaces, via a VPN
for VPNed interfaces, via a Proxy for proxied interfaces or via a
Relay for native interfaces) and waits for an RA message reply (see
Section 3.15.3). 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 Server and try another Server. Otherwise, the Client
processes the PD information found in the RA message.

Next, the Client creates a symmetric neighbor cache entry with the
Server's AERO address as the network-layer address and the address in
the first SLLAO as the Server's INET address. The Client records the
RA Router Lifetime field value in the neighbor cache entry as the
time for which the Server has committed to maintaining the MNP in the
routing system. The Client then autoconfigures AERO addresses for
each of the delegated MNPs and assigns them to the AERO interface.
The Client also caches any MSPs included in Route Information Options
(RIOs) [RFC4191] as MSPs to associate with the AERO link, and assigns
the MTU value in the MTU option to its AERO interface while
configuring an appropriate MRU.

The Client then registers additional ANET interfaces with the Server
by sending RS messages via each additional ANET interface. The RS
messages include the same parameters as for the initial RS/RA
exchange, but with destination address set to the Server's AERO
address and with an SLLAO specific to the ANET interface.

The Client examines the X and N bits in the SLLAO with S set to 1 in
each RA message it receives. If X is 1 the Client infers that there
is a Proxy on the path, and if N is 1 the Client infers that there is
a NAT on the path. If N is 1, the Client SHOULD set Port Number and
Link-Layer Address to 0 in the first S/TLLAO of any subsequent ND
messages it sends to the Server over that link.

Following autoconfiguration, the Client 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
subsequently maintains its MNP delegations through each of its
Servers by sending additional RS messages before Router Lifetime
expires.

After the Client registers its ANET interfaces, it may wish to change
one or more registrations, e.g., if an ANET 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 ANET
interface. The RS MUST include an SLLAO with S set to 1 for the
selected ANET interface and MAY include any additional SLLAOs
specific to other ANET interfaces. The Client includes fresh P(i)
values in each SLLAO to update the Server's neighbor cache entry. If
the Client wishes to update only the P(i) values, it sets the Port
Number and Link-Layer Address fields to 0. If the Client wishes to
disable the underlying interface, it sets D to 1. When the Client
receives the Server's RA response, it has assurance that the Server
has been updated with the new information.

If the Client wishes to discontinue use of a Server it issues an RS
message over any underlying interface with an SLLAO with R set to 1 .
When the Server processes the message, it releases the MNP, sets the
symmetric neighbor cache entry state for the Client to DEPARTED,
withdraws the IP route from the routing system and returns an RA
reply with Router Lifetime set to 0.

Clients SHOULD NOT remain associated with multiple Servers for long
durations, since routing inconsistencies could cause some fragments
of a fragmented packet to be delivered to Server A and others to
Server B. Clients SHOULD therefore associate with a single primary
Server, and send a departure message to the former Server soon after
moving to a new Server.

3.15.3. **AERO Server Behavior**

   AERO Servers act as IP routers and support a PD service for Clients.
   Servers arrange to add their AERO and INET addresses to a static map
   of Server addresses for the link and/or the DNS resource records for
   the FQDN "linkupnetworks.[domainname]" before entering service. The
   list of Server addresses should be geographically and/or
   topologically referenced, and forms the MAP list for the AERO link.

   When a Server receives a prospective Client's RS message on its AERO
   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 Server authenticates the RS
   message and processes the PD parameters. The Server first determines
   the correct MNPs to delegate to the Client by searching the Client
   database. When the Server delegates the MNPs, it also creates an IP
   forwarding table entry for each MNP so that the MNPs are propagated
   into the routing system (see: Section 3.3). For IPv6, the Server
   creates a single IPv6 forwarding table entry for each MNP. For IPv4,
   the Server creates both an IPv4 forwarding table entry and an IPv6
   forwarding table entry with the IPv4-mapped IPv6 address
   corresponding to the IPv4 address.

   The Server next creates a symmetric neighbor cache entry for the
   Client using the base AERO address as the network-layer address and
   with lifetime set to no more than the smallest PD lifetime. Next,
   the Server updates the neighbor cache entry by recording the
   information in each SLLAO in the RS indexed by the Interface ID and
   including the Port Number, Link Layer Address and P(i) values. For
   the SLLAO with S set to 1, however, the Server records the actual
   INET header source addresses instead of those that appear in the
   SLLAO in case there was a NAT in the path. The Server also records
   the value of the X bit to indicate whether there is a Proxy on the
   path.

   Next, the Server prepares an RA message using its AERO address as the
   network-layer source address and the network-layer source address of
   the RS message as the network-layer destination address. The Server
   includes the delegated MNPs, any other PD parameters and an SLLAO
   with the Link Layer Address set to the Server's SPAN address and with
   Interface ID set to 0xffff. The Server then includes one or more
   RIOs that encode the MSPs for the AERO link, plus an MTU option for
   the link MTU (see Section 3.13). The Server finally forwards the
   message to the Client using SPAN, INET or NULL encapsulation
   according to the Client interface type. (For Proxy/Native
   interfaces, the Server encapsulates the message in a SPAN header with
   source address set to its own SPAN address and destination address

set to the Proxy's (or Client's) SPAN address, then forwards the
message into the SPAN.)

After the initial RS/RA exchange, the Server maintains the symmetric
neighbor cache entry for the Client. If the Client (or Proxy) issues
additional NS/RS messages, the Server resets ReachableTime. If the
Client (or Proxy) issues an RS with PD release parameters (e.g., by
including an SLLAO with R set to 1), or if the Client becomes
unreachable, the Server sets the Client's symmetric neighbor cache
entry to the DEPARTED state and withdraws the IP routes from the AERO
routing system.

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

3.15.3.1. Lightweight DHCPv6 Relay Agent (LDRA)

When DHCPv6 is used as the ND/PD service back end, AERO Clients and
Servers are always on the same link (i.e., the AERO link) from the
perspective of DHCPv6. However, in some implementations the DHCPv6
server and ND function may be located in separate modules. In that
case, the Server's AERO interface module can act as a Lightweight
DHCPv6 Relay Agent (LDRA)[RFC6221] to relay PD messages to and from
the DHCPv6 server module.

When the LDRA receives an authentic RS message, it extracts the PD
message parameters and uses them to construct an IPv6/UDP/DHCPv6
message. It sets the IPv6 source address to the source address of
the RS message, sets the IPv6 destination address to
'All_DHCP_Relay_Agents_and_Servers' and sets the UDP fields to values
that will be understood by the DHCPv6 server.

The LDRA then wraps the message in a DHCPv6 'Relay-Forward' message
header and includes an 'Interface-Id' option that includes enough
information to allow the LDRA to forward the resulting Reply message
back to the Client (e.g., the Client's link-layer addresses, a
security association identifier, etc.). The LDRA also wraps the
information in all of the SLLAOs from the RS message into the
Interface-Id option, then forwards the message to the DHCPv6 server.

When the DHCPv6 server prepares a Reply message, it wraps the message
in a 'Relay-Reply' message and echoes the Interface-Id option. The
DHCPv6 server then delivers the Relay-Reply message to the LDRA,

which discards the Relay-Reply wrapper and IPv6/UDP headers, then
uses the DHCPv6 message to construct an RA response to the Client.
The Server uses the information in the Interface-Id option to prepare
the RA message and to cache the link-layer addresses taken from the
SLLAOs echoed in the Interface-Id option.

3.16. The AERO Proxy

Clients may connect to ANETs that require a perimeter security
gateway to enable communications to 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 Server's
   AERO address, the Proxy proceeds to the next step. Otherwise, if
   the destination is all-routers multicast the Proxy selects a
   "nearby" Server that is likely to be a good candidate to serve the
   Client and replaces the destination address with the Server's AERO
   address. Next, the Proxy creates a proxy neighbor cache entry and
   caches the Client and Server addresses along with any identifying
   information including Transaction IDs, Client Identifiers, Nonce
   values, etc. The Proxy then examines the address in the RS
   message SLLAO with S set to 1. If the address is different than
   the Client's ANET address, the Proxy notes that the Client is
   behind a NAT. The Proxy then sets X to 1 and changes the Link
   Layer Address to its own SPAN address. The Proxy finally
   encapsulates the RS message in a SPAN header with destination set
   to the Server's SPAN address then forwards the message into the
   SPAN.

o when the Server receives the RS message, it authenticates the
   message then creates or updates a symmetric neighbor cache entry
   for the Client with the Proxy's SPAN address as the link-layer
   address. The Server then sends an RA message with a single SLLAO
   back to the Proxy via the SPAN.

o when the Proxy receives the RA message, it matches the message
   with the RS that created the proxy neighbor cache entry. The
   Proxy then caches the PD route information as a mapping from the
   Client's MNPs to the Client's ANET address, and sets the neighbor
   cache entry state to REACHABLE. The Proxy then changes the SLLAO
   Link Layer Address to its own ANET address, sets X to 1, sets N to
   1 if the Client is behind a NAT, then re-encapsulates the RA
   message in an ANET header and forwards it to the Client.

After the initial RS/RA exchange, the Proxy forwards any Client data
packets for which there is no matching asymmetric neighbor cache
entry to the Server via the SPAN. Finally, the Proxy forwards any
Client data destined to an asymmetric neighbor cache target directly
to the target according to the link-layer information - the process
of establishing asymmetric neighbor cache entries is specified in
[Section 3.17](#).

While the Client is still attached to the ANET, the Proxy
periodically sends NS/RS messages to update the Server's symmetric
neighbor cache entries on behalf of the Client and/or to convey QoS
updates. If the Server ceases to send solicited NA/RA responses, the
Proxy marks the Server as unreachable and sends an unsolicited RA
with Router Lifetime set to zero to inform the Client that this
Server is no longer able to provide Service. If the Client becomes
unreachable, the Proxy sets the neighbor cache entry state to
DEPARTED and sends an RS message to the Server with an SLLAO with D
set to 1 and with Interface ID set to the Client's interface ID so
that the Server will de-register this Interface ID. Although the
Proxy engages in these 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.

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 data
messages without encapsulation so that the ANET native routing system
transports the unencapsulated 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 AERO-aware, the Client can
avoid 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 AERO address and with destination
address set to the AERO address of the Client's selected Server or to
all-routers multicast. The Client includes an SLLAO with Interface
ID, Prefix Length and P(i) information but with Port Number and Link-
Layer Address set to 0.

The Client then sends the unencapsulated RS message, which will be
intercepted by the AERO-Aware access router. The access router then
encapsulates the RS message in an ANET header with its own address as
the source address and the address of a Proxy as the destination
address. The access router further remembers the address of the
Proxy so that it can encapsulate future data 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 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 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.

3.17. AERO Route Optimization

While data 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, NATed and Direct interfaces, the Server is
   the ROS.

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

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

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

The route optimization procedure is conducted between the ROS and the
target Server/Gateway 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 Server, or a non-MNP correspondent reachable via a Gateway.

The procedures are specified in the following sections.

3.17.1. Route Optimization Initiation

While data packets are flowing from the source node toward a target
node, the ROS performs address resolution by sending an NS message to
receive a solicited NA message from the ROR.

When the ROS sends an NS, it includes the AERO address of the ROS as
the source address (e.g., fe80::1) and the AERO address corresponding
to the data packet's destination address as the destination address
(e.g., if the destination address is 2001:db8:1:2::1 then the
corresponding AERO address is fe80::2001:db8:1:2). The NS message
includes an SLLAO with Link Layer Address set to the SPAN address of
the ROS and with all other fields set to 0. The message SHOULD also
include a Nonce and Timestamp option if the ROS needs to correlate NA
replies.

The ROS then encapsulates the NS message in a SPAN header with source
set to its own SPAN address and destination set to the data packet's
destination address, then sends it into the SPAN without decrementing
the network-layer TTL/Hop Limit field.

3.17.2. Relaying the NS

When the Relay receives the NS message from the ROS, it discards the
INET header and determines that the ROR is the next hop by consulting
its standard IPv6 forwarding table for the SPAN header destination
address. The Relay then forwards the SPAN message toward the ROR the
same as for any IPv6 router. The final-hop Relay in the SPAN will
deliver the message via a secured tunnel to the ROR.

3.17.3. Processing the NS and Sending the NA

When the ROR receives the NS message, it examines the AERO
destination address to determine whether it has a neighbor cache
entry and/or route that matches the target; if not, it drops the NS
message and returns from processing. Next, if the target belongs to
an MNP Client neighbor in the DEPARTED state the ROR changes the NS
message SPAN destination address to the address of the Client's new
Server, forwards the message into the SPAN and returns from
processing. If the target belongs to an MNP Client neighbor in the
REACHABLE state, the ROR instead adds the AERO source address to the
target Client's Report List with time set to ReportTime. If the
target belongs to a non-MNP route, the ROR continues processing
without adding an entry to the Report List.

The ROR then prepares a solicited NA message to send back to the ROS
but does not create a neighbor cache entry. The ROR sets the NA
source address to the destination AERO address of the NS, and
includes the Nonce value received in the NS plus the current
Timestamp. The ROR next includes a TLLAO with Interface ID set to
0xffff, with S set to 1, with all P(i) values set to "low", and with
Link Layer Address set to the ROR's SPAN address. If the target
belongs to an MNP Client, the ROR sets the Prefix Length to the MNP
prefix length; otherwise, it sets Prefix Length to the maximum of the
non-MNP prefix length and 64. (Note that a /64 limit is imposed to
avoid causing the ROS to set short prefixes (e.g., "default") that
would match destinations for which the routing system includes more-
specific prefixes. Note also that prefix lengths longer than /64 are
out of scope for this specification.)

If the target belongs to an MNP Client, the ROR next includes
additional TLLAOs for all of the target Client's Interface IDs. For
NATed, VPNed and Direct interfaces, the TLLAO Link Layer Addresses
are the SPAN address of the ROR. For Proxyed and native interfaces,

the TLLAO Link Layer Addresses are the SPAN addresses of the Proxys
and the Client's native interfaces. The ROR finally encapsulates the
NA message in a SPAN header with source set to its own SPAN address
and destination set to the source SPAN address of the NS message,
then forwards the message into the SPAN without decrementing the
network-layer TTL/Hop Limit field.

3.17.4. Relaying the NA

When the Relay receives the NA message from the ROR, it discards the
INET header and determines that the ROS is the next hop by consulting
its standard IPv6 forwarding table for the SPAN header destination
address. The Relay then forwards the SPAN-encapsulated NA message
toward the ROS the same as for any IPv6 router. The final-hop Relay
in the SPAN will deliver the message via a secured tunnel to the ROS.

3.17.5. Processing the NA

When the ROS receives the solicited NA message, it discards the INET
and SPAN headers. The ROS next verifies the Nonce and Timestamp
values, then creates an asymmetric neighbor cache entry for the ROR
and caches all information found in the solicited NA TLLAOs. The ROS
finally sets the asymmetric neighbor cache entry lifetime to
ReachableTime seconds.

3.17.6. Route Optimization Maintenance

Following route optimization, the ROS forwards future data packets
destined to the target via the addresses found in the cached link-
layer information. The route optimization is shared by all sources
that send packets to the target via the ROS, i.e., and not just the
source on behalf of which the route optimization was initiated.

While new data packets destined to the target are flowing through the
ROS, it sends additional NS messages to the ROR before ReachableTime
expires to receive a fresh solicited NA 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 C.1).

The ROS then updates the asymmetric neighbor cache entry to refresh
ReachableTime, while (for MNP destinations) the ROR adds or updates
the ROS address to the target Client's Report List and with time set
to ReportTime. While no data packets are flowing, the ROS instead
allows ReachableTime for the asymmetric neighbor cache entry to
expire. When ReachableTime expires, the ROS deletes the asymmetric
neighbor cache entry. Future data packets flowing through the ROS

will again trigger a new route optimization exchange while initial
data packets travel over a suboptimal route.

The ROS may also receive unsolicited NA messages from the ROR at any
time. If there is an asymmetric 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 the forward path is still working. If there is no asymmetric
neighbor cache entry, the route optimization source simply discards
the unsolicited NA. Cases in which unsolicited NA messages are
generated are specified in Section 3.19.

In this arrangement, the ROS holds an asymmetric neighbor cache entry
for the ROR, but the ROR does not hold an asymmetric neighbor cache
entry for the ROS. The route optimization neighbor relationship is
therefore asymmetric and unidirectional. If the target node also has
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.

3.18. Neighbor Unreachability Detection (NUD)

AERO nodes perform Neighbor Unreachability Detection (NUD) per
[RFC4861]. NUD is performed either reactively in response to
persistent link-layer errors (see Section 3.14) or proactively to
confirm reachability. The NUD algorithm may further be seeded by ND
hints of forward progress, but care must be taken to avoid inferring
reachability based on spoofed information.

When an ROR directs an ROS to a neighbor with one or more target
link-layer addresses, the ROS can proactively test each direct path
by sending an initial NS message to elicit a solicited NA response.
While testing the paths, the ROS can optionally continue sending
packets via the SPAN, maintain a small queue of packets until target
reachability is confirmed, or (optimistically) allow packets to flow
via the direct paths. In any case, the ROS should only consider the
neighbor unreachable if NUD fails over multiple target link-layer
address paths.

When a ROS sends an NS message used for NUD, it uses its AERO
addresses as the IPv6 source address and the AERO address
corresponding to a target link-layer address as the destination. For
each target link-layer address, the source node encapsulates the NS
message in SPAN/INET headers with its own SPAN address as the source
and the SPAN address of the target as the destination, If the target
is located within the same SPAN segment, the source sets the INET
address of the target as the destination; otherwise, it sets the INET

address of a Relay as the destination. The source then forwards the
message into the SPAN.

Paths that pass NUD tests are marked as "reachable", while those that
do not are marked as "unreachable". These markings inform the AERO
interface forwarding algorithm specified in Section 3.12.

Proxys can perform NUD to verify Server reachability on behalf of
their proxied Clients so that the Clients need not engage in NUD
messaging themselves.

3.19. Mobility Management and Quality of Service (QoS)

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

Servers provide a Mobility Anchor Point (MAP) for their dependent
Clients. Clients are responsible for maintaining neighbor
relationships with their 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 Server with this new
information. Note that for Proxied interfaces, however, the Proxy
can perform the RS/RA exchanges on the Client's behalf.

Mobility management considerations are specified in the following
sections.

3.19.1. Mobility Update Messaging

Servers acting as MAPs accommodate mobility and/or QoS change events
by sending an unsolicited NA message to each ROS in the target
Client's Report List. When a MAP sends an unsolicited NA message, it
sets the IPv6 source address to the Client's AERO address and sets
the IPv6 destination address to all-nodes multicast (ff02::1). The
MAP also includes a TLLAO with Interface ID 0xffff, S set to 1 and
Link Layer address set to the MAP's SPAN address, and includes
additional TLLAOs for all of the target Client's Interface IDs with
Link Layer Addresses set to the corresponding SPAN addresses. The
MAP finally encapsulates the message in a SPAN header with source set
to its own SPAN address and destination set to the SPAN address of
the ROS, then sends the message to a Relay in the SPAN.

As for the hot-swap of interface cards discussed in Section 7.2.6 of [RFC4861], the transmission and reception of unsolicited NA messages is unreliable but provides a useful optimization. In well-connected Internetworks with robust data links unsolicited NA messages will be delivered with high probability, but in any case the MAP can optionally send up to MAX_NEIGHBOR_ADVERTISEMENT unsolicited NAs to each ROS to increase the likelihood that at least one will be received.

When an ROS receives an unsolicited NA message, it ignores the message if there is no existing neighbor cache entry for the Client. Otherwise, it uses the included TLLAOs to update the Link Layer Address and QoS information in the neighbor cache entry, but does not reset ReachableTime since the receipt of an unsolicited NA message from the target Server does not provide confirmation that any forward paths to the target Client are working.

If unsolicited NA 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 packets to the target Client according to its current neighbor cache information but may receive persistent unsolicited NA messages as discussed in Section 3.19.5.1.

3.19.2. Announcing Link-Layer Address and/or QoS Preference Changes

When a Client needs to change its ANET addresses and/or QoS preferences (e.g., due to a mobility event), either the Client or its Proxys send RS messages to the Server via the SPAN with SLLAOs that include the new Client Port Number, Link Layer Address and P(i) values. If the RS messages are sent solely for the purpose of updating QoS preferences, Port Number and Link-Layer Address are set to 0.

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

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

3.19.3. Bringing New Links Into Service

When a Client needs to bring new ANET interfaces into service (e.g., when it activates a new data link), it sends an RS message to its Server via the ANET interface with SLLAOs that include the new Client Link Layer Address information.

3.19.4. Removing Existing Links from Service

   When a Client needs to remove existing ANET interfaces from service
   (e.g., when it de-activates an existing data link), it sends an RS
   message to its Server with SLLAOs with D set to 1.

   If the Client needs to send RS messages over an ANET interface other
   than the one being removed from service, it MUST include a current
   SLLAO with S set to 1 for the sending interface and include
   additional SLLAOs with S set to 0 for any ANET interfaces being
   removed from service.

3.19.5. Moving to a New Server

   When a Client associates with a new Server, it performs the Client
   procedures specified in Section 3.15.2. The Client then sends an RS
   message over any working ANET interface with destination set to the
   old Server's AERO address and with an SLLAO with R set to 1 to fully
   release itself from the old Server. The SLLAO also includes the SPAN
   address of the new Server in the Link Layer Address. If the Client
   does not receive an RA reply after MAX_RTR_SOLICITATIONS attempts
   over multiple ANET interfaces, the old Server may have failed and the
   Client should discontinue its release attempts.

   When the old Server processes the RS, it sends unsolicited NA
   messages with a single TLLAO with Interface ID set to 0xffff and with
   R and S set to 1 to all ROSs in the Client's Report List. The Server
   also changes the symmetric neighbor cache entry state to DEPARTED,
   sets the link-layer address of the Client to the address found in the
   RS SLLAO (i.e., the SPAN address of the new Server), and sets a timer
   to DepartTime seconds. The old Server then returns an immediate RA
   message to the Client with Router Lifetime set to 0. After a short
   delay (e.g., 2 seconds), the old Server withdraws the Client's MNP
   from the routing system. After DepartTime expires, the old Server
   deletes the symmetric neighbor cache entry.

   Clients SHOULD NOT move rapidly between 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 Server include a
   Server that has gone unreachable, topological movements of
   significant distance, movement to a new geographic region, movement
   to a new SPAN segment, etc.

3.19.5.1. Accommodating Orphaned Fragments

   When a Client moves to a new Server, there is a chance that some of
   the fragments of a multiple fragment SPAN packet have already arrived
   at the old Server, while others are en route to the new Server. The

old Server forwards these orphaned fragments to the new Server over
the SPAN by encapsulating them in a second SPAN header with the old
Server's SPAN address as the source and the SPAN address of the new
Server as the destination.

The new Server will then receive a packet that has an outer SPAN
header inserted by the old Server and an inner SPAN header inserted
by the original ROS. It then discards the outer SPAN header and adds
the enclosed fragment to the reassembly buffer for this SPAN packet.
If the Client has already departed from this new Server, the new
Server instead re-encapsulates the fragment by replacing the outer
SPAN header and forwarding the packet to the next Server the same as
described above. In this way, wayward fragments will follow the
trail of departed Servers until they eventually reach the current
Server.

Note that in some cases reassemblies may be left incomplete after the
Client has departed. Proxys and Servers discard any incomplete
reassemblies when the DEPARTED neighbor cache entry is deleted.

3.20. 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
ANET 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 Gateways 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.

3.20.1. Source-Specific Multicast (SSM)

When an ROS (i.e., an AERO Proxy/Server/Gateway) "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-AERO
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 AERO interface, X
originates a separate copy of the Join/Prune for each (S,G) in the

message using its own AERO address as the source address and ALL-PIM-
ROUTERS as the destination address. X then encapsulates each message
in a SPAN header with source address set to the SPAN address of X and
destination address set to S then forwards the message into the SPAN.
The SPAN in turn forwards the message to AERO Server/Gateway "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 3.17. The resulting NAs will return the AERO address for the
prefix that matches S as the network-layer source address and TLLAOs
with the SPAN addresses corresponding to any Interface IDs that are
currently servicing S.

When Y processes the Join/Prune message, if S located behind any
Native, Direct, VPNed or NATed 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 SPAN while continuing to use the AERO address of X
as the source address. Each Z* then updates its MRIB accordingly and
maintains the AERO address of X as the next hop in the reverse path.
Since the Relays in the SPAN 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 SPAN segment
as X, the multicast data traffic sent to X directly using SPAN/INET
encapsulation instead of via a Relay.

Following the initial Join/Prune and NS/NA messaging, X maintains an
asymmetric 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 SPAN.

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 3.19. When X receives the
unsolicited NA message, it updates its asymmetric neighbor cache
entry for the AERO address 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 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 AERO address as the
source address. This causes Y2 to include Y1 in the multicast
forwarding tree during the interim time that Y1's symmetric neighbor
cache entry for C is in the DEPARTED state. At the same time, Y1
sends an unsolicited NA message to X with an Interface ID 0xffff and
R set to 1 to cause X to release its asymmetric neighbor cache entry.
X then sends a new Join message to S via the SPAN and re-initiates
route optimization the same as if it were receiving a fresh Join
message from a node on a downstream link.

3.20.2. 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 SPAN. X uses its own AERO address as the source
address and ALL-PIM-ROUTERS as the destination address, then
encapsulates each message in a SPAN header with source address set to
the SPAN address of X and destination address set to R, then sends
the message into the SPAN. 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 3.20.1.

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
packets in PIM Register messages and forwards them to R via the SPAN.
R may then elect to send a PIM Join to Z* over the SPAN. 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 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 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 3.20.1. Once the (S,G) tree is
established, the listeners can send (S, G) Prune messages to R so
that multicast 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.

3.20.3. 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.

3.21. Operation over Multiple AERO Links (VLANs)

An AERO Client can connect to multiple AERO links the same as for any
data link service. In that case, the Client maintains a distinct
AERO interface for each link, e.g., 'aero0' for the first link,
'aero1' for the second, 'aero2' for the third, etc. Each AERO link
would include its own distinct set of Relays, Servers and Proxys,
thereby providing redundancy in case of failures.

The Relays, Servers and Proxys on each AERO link can assign AERO and
SPAN addresses that use the same or different numberings from those
on other links. Since the links are mutually independent there is no
requirement for avoiding inter-link address duplication, e.g., the
same AERO address such as fe80::1000 could be used to number distinct
nodes that connect to different links.

Each AERO link could utilize the same or different ANET connections.
The links can be distinguished at the link-layer via 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,
where each VLAN is distinguished by a different label (e.g., colors
such as Red, Green, Blue, etc.). In particular, the Client can tag
its RS messages with the appropriate label to cause the network to
select the desired VLAN.

Clients that connect to multiple AERO interfaces can select the
outgoing interface appropriate for a given Red/Blue/Green/etc.
traffic profile while (in the reverse direction) correspondent nodes
must have some way of steering their packets destined to a target via
the correct AERO link.

In a first alternative, if each AERO 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 Red/Green/Blue/etc.
network 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
Relays.

In a second alternative, if each AERO link services the same MSP(s)
then each link could assign a distinct "AERO Link Anycast" address
that is configured by all Relays on the link. Correspondent nodes
then include a "type 4" routing header with the Anycast address for
the AERO link as the IPv6 destination and with the address of the
target encoded as the "next segment" in the routing header
[RFC8402][I-D.ietf-6man-segment-routing-header]. Standard IP routing
will then direct the packet to the nearest Relay for the correct AERO
link, which will replace the destination address with the target
address then forward the packet to the target.

3.22. 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 Gateway NAT64 mapping caches. In that way, an IPv4
correspondent node can send packets to the IPv4 address mapping of
the target MN, and the Gateway 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 Server, the Server returns
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.

3.23. Transition Considerations

The SPAN ensures that dissimilar INET partitions can be joined into a
single unified AERO 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 SPAN
segments. This can be accomplished by incrementally deploying AERO
Gateways on each INET partition, with each Gateway distributing its
MNPs and/or discovering non-MNP 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 AERO link view
(bridged by the SPAN) even if the INET partitions remain in their
current protocol and addressing plans. In that way, the AERO link
can serve the dual purpose of providing a mobility 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 AERO link MNP-based addressing scheme, the partition and
AERO link can be joined by Gateways.

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

4. Implementation Status

An 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. The latest
versions are available at: http://linkupnetworks.net/aero.

5. IANA Considerations

The IANA has assigned a 4-octet Private Enterprise Number "45282" for
AERO in the "enterprise-numbers" registry.

The IANA has assigned the UDP port number "8060" for an earlier
experimental version of AERO [RFC6706]. This document obsoletes
[RFC6706] and claims the UDP port number "8060" for all future use.

No further IANA actions are required.

6. Security Considerations

AERO Relays configure secured tunnels with AERO Servers and Proxys
within their local SPAN segments. Applicable secured tunnel
alternatives include IPsec [RFC4301], TLS/SSL [RFC8446], DTLS
[RFC6347], etc. The AERO Relays of all SPAN segments in turn
configure secured tunnels for their neighboring AERO Relays across
the SPAN. Therefore, packets that traverse the SPAN between any pair
of AERO link neighbors are already secured.

AERO Servers, Gateways and Proxys targeted by a route optimization
may also receive packets directly from the INET partitions instead of
via the SPAN. For INET partitions that apply effective ingress
filtering to defeat source address spoofing, the simple data origin
authentication procedures in Section 3.11 can be applied. This
implies that the ROS list must be maintained consistently by all
route optimization targets within the same INET partition, and that
the ROS list must be securely managed by the partition's
administrative authority.

For INET partitions that cannot apply effective ingress filtering,
the two options for securing communications include 1) disable route

optimization so that all traffic is conveyed over secured tunnels via
the SPAN, 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 overburdened 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 enclaves need not apply security
to their ND messages, since the messages will be intercepted by a
perimeter Proxy that applies security on its outward-facing
interface. AERO Clients located outside of secured enclaves SHOULD
use symmetric network and/or transport layer security services, but
when there are many prospective neighbors with dynamically changing
connectivity an asymmetric security service such as SEND may be
needed (see: Appendix C.6).

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 symmetric network
and/or transport layer security services such as IPsec, TLS/SSL,
DTLS, etc. AERO Proxys and 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 Servers and Relays 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 Servers and Relays 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 Servers and Proxys can institute rate limits that
protect Clients from receiving packet floods that could DoS low data
rate links.

AERO Gateways must implement ingress filtering to avoid a spoofing
attack in which spurious SPAN messages are injected into an AERO 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 and ROS lists 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, e.g.,
through secure upload of a static file, through DNS lookups, etc.
The ROS list can be conveyed to Servers and Proxys through
administrative action, secured file distribution, etc.

Although public domain and commercial SEND implementations exist,
concerns regarding the strength of the cryptographic hash algorithm
have been documented [RFC6273] [RFC4982].

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

7. Acknowledgements

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 Information Technology (BIT)
MobileNet program.

This work is aligned with the Boeing autonomy program.

8. References

8.1. Normative References

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

[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>.

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

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

   [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>.

   [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>.

   [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.ietf-6man-segment-routing-header]
            Filsfils, C., Dukes, D., Previdi, S., Leddy, J.,
            Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment
            Routing Header (SRH)", draft-ietf-6man-segment-routing-
            header-19 (work in progress), May 2019.

[I-D.ietf-dmm-distributed-mobility-anchoring]
            Chan, A., Wei, X., Lee, J., Jeon, S., and C. Bernardos,
            "Distributed Mobility Anchoring", draft-ietf-dmm-
            distributed-mobility-anchoring-13 (work in progress),
            March 2019.

[I-D.ietf-intarea-gue]
            Herbert, T., Yong, L., and O. Zia, "Generic UDP
            Encapsulation", draft-ietf-intarea-gue-07 (work in
            progress), March 2019.

[I-D.ietf-intarea-gue-extensions]
            Herbert, T., Yong, L., and F. Templin, "Extensions for
            Generic UDP Encapsulation", draft-ietf-intarea-gue-
            extensions-06 (work in progress), March 2019.

[I-D.ietf-intarea-tunnels]
            Touch, J. and M. Townsley, "IP Tunnels in the Internet
            Architecture", draft-ietf-intarea-tunnels-09 (work in
            progress), July 2018.

[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-02 (work in progress), May 2019.

[I-D.templin-6man-dhcpv6-ndopt]
            Templin, F., "A Unified Stateful/Stateless Configuration
            Service for IPv6", draft-templin-6man-dhcpv6-ndopt-07
            (work in progress), December 2018.

[I-D.templin-intarea-grefrag]
            Templin, F., "GRE Tunnel Level Fragmentation", draft-
            templin-intarea-grefrag-04 (work in progress), July 2016.

[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-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-23 (work in progress),
          December 2018.

[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>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -
          Communication Layers", STD 3, RFC 1122,
          DOI 10.17487/RFC1122, October 1989,
          <https://www.rfc-editor.org/info/rfc1122>.

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

[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>.

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

   [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>.

   [RFC2764] Gleeson, B., Lin, A., Heinanen, J., Armitage, G., and A.
              Malis, "A Framework for IP Based Virtual Private
              Networks", RFC 2764, DOI 10.17487/RFC2764, February 2000,
              <https://www.rfc-editor.org/info/rfc2764>.

   [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P.
              Traina, "Generic Routing Encapsulation (GRE)", RFC 2784,
              DOI 10.17487/RFC2784, March 2000,
              <https://www.rfc-editor.org/info/rfc2784>.

   [RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE",
              RFC 2890, DOI 10.17487/RFC2890, September 2000,
              <https://www.rfc-editor.org/info/rfc2890>.

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

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

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

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

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

   [RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms
              for IPv6 Hosts and Routers", RFC 4213,
              DOI 10.17487/RFC4213, October 2005,
              <https://www.rfc-editor.org/info/rfc4213>.

   [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>.

   [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for
              IP", RFC 4607, DOI 10.17487/RFC4607, August 2006,
              <https://www.rfc-editor.org/info/rfc4607>.

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

   [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>.

   [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>.

   [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>.

   [RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field",
              RFC 6864, DOI 10.17487/RFC6864, February 2013,
              <https://www.rfc-editor.org/info/rfc6864>.

   [RFC7269] Chen, G., Cao, Z., Xie, C., and D. Binet, "NAT64
              Deployment Options and Experience", RFC 7269,
              DOI 10.17487/RFC7269, June 2014,
              <https://www.rfc-editor.org/info/rfc7269>.

   [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>.

   [RFC8086] Yong, L., Ed., Crabbe, E., Xu, X., and T. Herbert, "GRE-
              in-UDP Encapsulation", RFC 8086, DOI 10.17487/RFC8086,
              March 2017, <https://www.rfc-editor.org/info/rfc8086>.

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

   [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>.

Appendix A. AERO Alternate Encapsulations

   When GUE encapsulation is not needed, AERO can use common
   encapsulations such as IP-in-IP [RFC2003][RFC2473][RFC4213], Generic
   Routing Encapsulation (GRE) [RFC2784][RFC2890] and others. The
   encapsulation is therefore only differentiated from non-AERO tunnels
   through the application of AERO control messaging and not through,
   e.g., a well-known UDP port number.

   As for GUE encapsulation, alternate AERO encapsulation formats may
   require encapsulation layer fragmentation. For simple IP-in-IP
   encapsulation, an IPv6 fragment header is inserted directly between
   the inner and outer IP headers when needed, i.e., even if the outer
   header is IPv4. The IPv6 Fragment Header is identified to the outer
   IP layer by its IP protocol number, and the Next Header field in the
   IPv6 Fragment Header identifies the inner IP header version. For GRE
   encapsulation, a GRE fragment header is inserted within the GRE
   header [I-D.templin-intarea-grefrag].

   Figure 6 shows the AERO IP-in-IP encapsulation format before any
   fragmentation is applied:

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Outer IPv4 Header     |        |     Outer IPv6 Header     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |IPv6 Frag Header (optional)|        |IPv6 Frag Header (optional)|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      Inner IP Header      |        |      Inner IP Header      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          |        |                           |
   ~                          ~        ~                           ~
   ~     Inner Packet Body    ~        ~     Inner Packet Body     ~
   ~                          ~        ~                           ~
   |                          |        |                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   Minimal Encapsulation in IPv4     Minimal Encapsulation in IPv6
```
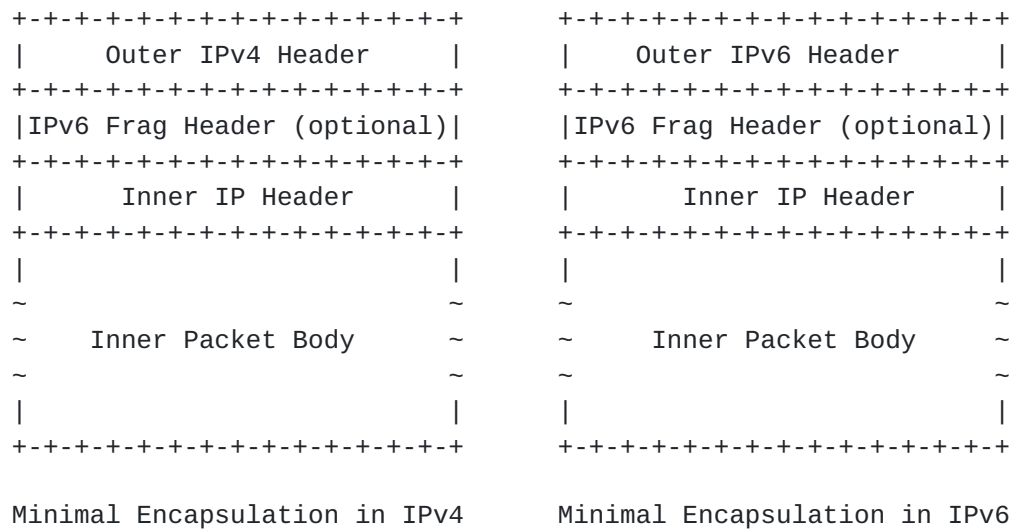
Figure 6: Minimal Encapsulation Format using IP-in-IP

Figure 7 shows the AERO GRE encapsulation format before any
fragmentation is applied:

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        Outer IP Header      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          GRE Header         |
   | (with checksum, key, etc..) |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | GRE Fragment Header (optional)|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Inner IP Header     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             |
   ~                             ~
   ~        Inner Packet Body    ~
   ~                             ~
   |                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Minimal Encapsulation Using GRE

Alternate encapsulation may be preferred in environments where GUE
encapsulation would add unnecessary overhead. For example, certain
low-bandwidth wireless data links may benefit from a reduced
encapsulation overhead.

GUE encapsulation can traverse network paths that are inaccessible to non-UDP encapsulations, e.g., for crossing Network Address Translators (NATs). More and more, network middleboxes are also being configured to discard packets that include anything other than a well-known IP protocol such as UDP and TCP. It may therefore be necessary to determine the potential for middlebox filtering before enabling alternate encapsulation in a given environment.

In addition to IP-in-IP, GRE and GUE, AERO can also use security encapsulations such as IPsec, TLS/SSL, DTLS, etc. In that case, AERO control messaging and route determination occur before security encapsulation is applied for outgoing packets and after security decapsulation is applied for incoming packets.
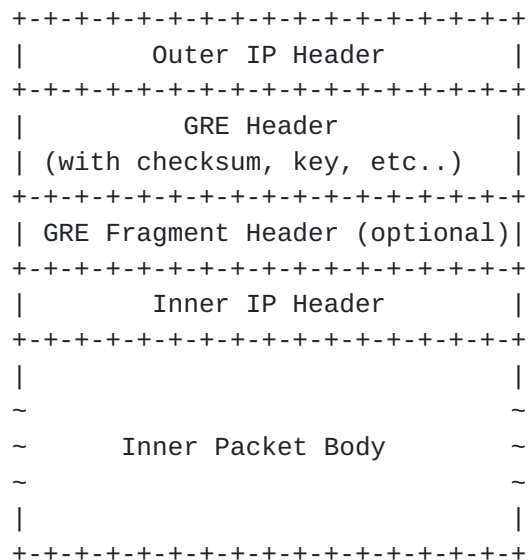
AERO is especially well suited for use with VPN system encapsulations such as OpenVPN [OVPN].

Appendix B. S/TLLAO Extensions for Special-Purpose Links

The AERO S/TLLAO format specified in Section 3.6 includes a Length value of 5 (i.e., 5 units of 8 octets). However, special-purpose links may extend the basic format to include additional fields and a Length value larger than 5.

For example, adaptation of AERO to the Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS) includes link selection preferences based on transport port numbers in addition to the existing DSCP-based preferences. ATN/IPS nodes maintain a map of transport port numbers to 64 possible preference fields, e.g., TCP port 22 maps to preference field 8, TCP port 443 maps to preference field 20, UDP port 8060 maps to preference field 34, etc. The extended S/TLLAO format for ATN/IPS is shown in Figure 8, where the Length value is 7 and the 'Q(i)' fields provide link preferences for the corresponding transport port number.
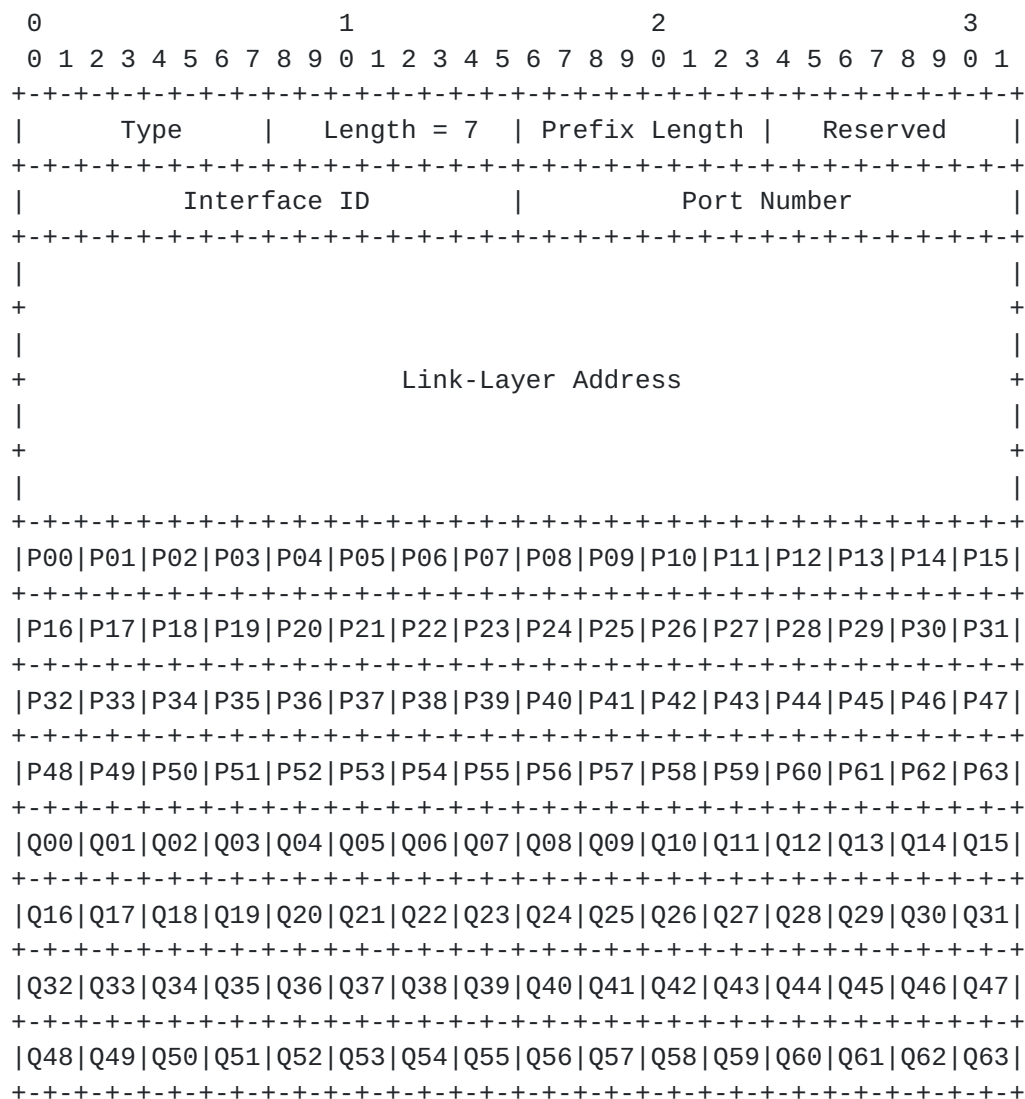
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |  Length = 7 | Prefix Length |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Interface ID       |           Port Number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                       Link-Layer Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P00|P01|P02|P03|P04|P05|P06|P07|P08|P09|P10|P11|P12|P13|P14|P15|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P16|P17|P18|P19|P20|P21|P22|P23|P24|P25|P26|P27|P28|P29|P30|P31|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P32|P33|P34|P35|P36|P37|P38|P39|P40|P41|P42|P43|P44|P45|P46|P47|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P48|P49|P50|P51|P52|P53|P54|P55|P56|P57|P58|P59|P60|P61|P62|P63|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Q00|Q01|Q02|Q03|Q04|Q05|Q06|Q07|Q08|Q09|Q10|Q11|Q12|Q13|Q14|Q15|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Q16|Q17|Q18|Q19|Q20|Q21|Q22|Q23|Q24|Q25|Q26|Q27|Q28|Q29|Q30|Q31|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Q32|Q33|Q34|Q35|Q36|Q37|Q38|Q39|Q40|Q41|Q42|Q43|Q44|Q45|Q46|Q47|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Q48|Q49|Q50|Q51|Q52|Q53|Q54|Q55|Q56|Q57|Q58|Q59|Q60|Q61|Q62|Q63|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: ATN/IPS Extended S/TLLAO Format

Appendix C. Non-Normative Considerations

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

C.1. Implementation Strategies for Route Optimization

Route optimization as discussed in Section 3.17 results in the route optimization source (ROS) creating an asymmetric neighbor cache entry for the target neighbor. The neighbor cache entry is maintained for at most REACHABLE_TIME 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
   to REACHABLE_TIME seconds 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 (REACHABLE_TIME - 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.

C.2. Implicit Mobility Management

   AERO 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 ANET 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 ANET 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.

C.3. Direct Underlying Interfaces

   When a Client's AERO 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, Servers and Relays in the
communications path. Direct interfaces must be tested periodically
for reachability, e.g., via NUD.

C.4. AERO Clients on the Open Internetwork

AERO Clients that connect to the open Internetwork via either a
native or NATed interface can establish a VPN to securely connect to
a Server. Alternatively, the Client can exchange ND messages
directly with other AERO nodes on the same SPAN segment using INET
encapsulation only and without joining the SPAN. In that case,
however, the Client must apply asymmetric security for ND messages to
ensure routing and neighbor cache integrity (see: Section 6).

C.5. Operation on AERO Links with /64 ASPs

IPv6 AERO links typically have MSPs that aggregate many candidate
MNPs of length /64 or shorter. However, in some cases it may be
desirable to use AERO over links that have only a /64 MSP. This can
be accommodated by treating all Clients on the AERO link as simple
hosts that receive /128 prefix delegations.

In that case, the Client sends an RS message to the Server the same
as for ordinary AERO links. The Server responds with an RA message
that includes one or more /128 prefixes (i.e., singleton addresses)
that include the /64 MSP prefix along with an interface identifier
portion to be assigned to the Client. The Client and Server then
configure their AERO addresses based on the interface identifier
portions of the /128s (i.e., the lower 64 bits) and not based on the
/64 prefix (i.e., the upper 64 bits).

For example, if the MSP for the host-only IPv6 AERO link is
2001:db8:1000:2000::/64, each Client will receive one or more /128
IPv6 prefix delegations such as 2001:db8:1000:2000::1/128,
2001:db8:1000:2000::2/128, etc. When the Client receives the prefix
delegations, it assigns the AERO addresses fe80::1, fe80::2, etc. to
the AERO interface, and assigns the global IPv6 addresses (i.e., the
/128s) to either the AERO interface or an internal virtual interface
such as a loopback. In this arrangement, the Client conducts route
optimization in the same sense as discussed in Section 3.17.

This specification has applicability for nodes that act as a Client
on an "upstream" AERO link, but also act as a Server on "downstream"
AERO links. More specifically, if the node acts as a Client to
receive a /64 prefix from the upstream AERO link it can then act as a
Server to provision /128s to Clients on downstream AERO links.

C.6. AERO Adaptations for SEcure Neighbor Discovery (SEND)

   SEcure Neighbor Discovery (SEND) [RFC3971] and Cryptographically
   Generated Addresses (CGAs) [RFC3972] were designed to secure IPv6 ND
   messaging in environments where symmetric network and/or transport-
   layer security services are impractical (see: Section 6). AERO nodes
   that use SEND/CGA employ the following adaptations.

   When a source AERO node prepares a SEND-protected ND message, it uses
   a link-local CGA as the IPv6 source address and writes the prefix
   embedded in its AERO address (i.e., instead of fe80::/64) in the CGA
   parameters Subnet Prefix field. When the neighbor receives the ND
   message, it first verifies the message checksum and SEND/CGA
   parameters while using the link-local prefix fe80::/64 (i.e., instead
   of the value in the Subnet Prefix field) to match against the IPv6
   source address of the ND message.

   The neighbor then derives the AERO address of the source by using the
   value in the Subnet Prefix field as the interface identifier of an
   AERO address. For example, if the Subnet Prefix field contains
   2001:db8:1:2, the neighbor constructs the AERO address as
   fe80::2001:db8:1:2. The neighbor then caches the AERO address in the
   neighbor cache entry it creates for the source, and uses the AERO
   address as the IPv6 destination address of any ND message replies.

C.7. AERO Critical Infrastructure Considerations

   AERO Relays can be either Commercial off-the Shelf (COTS) standard IP
   routers or virtual machines in the cloud. Relays must be
   provisioned, supported and managed by the INET administrative
   authority, and connected to the Relays of other INETs via inter-
   domain peerings. Cost for purchasing, configuring and managing
   Relays is nominal even for very large AERO links.

   AERO Servers can be standard dedicated server platforms, but most
   often will be deployed as virtual machines in the cloud. The only
   requirements for Servers are that they can run the AERO user-level
   code and have at least one network interface connection to the INET.
   As with Relays, Servers must be provisioned, supported and managed by
   the INET administrative authority. Cost for purchasing, configuring
   and managing Servers is nominal especially for virtual 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 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 Gateways can be any dedicated server or COTS router platform connected to INETs and/or EUNs. The Gateway joins the SPAN and engages in eBGP peering with one or more Relays as a stub AS. The Gateway then injects its MNPs and/or non-MNP prefixes into the BGP routing system, and provisions the prefixes to its downstream-attached networks. The Gateway can perform ROS and MAP services the same as for any Server, and can route between the MNP and non-MNP address spaces.

Appendix D. **Change Log**

<< RFC Editor - remove prior to publication >>

Changes from draft-templin-intarea-6706bis-14 to draft-templin-intrea-6706bis-15:

o MTU and fragmentation

o New details in movement to new Server

Changes from draft-templin-intarea-6706bis-13 to draft-templin-intrea-6706bis-14:

o Security based on secured tunnels, ingress filtering, MAP list and ROS list

Changes from draft-templin-intarea-6706bis-12 to draft-templin-intrea-6706bis-13:

o New paragraph in Section 3.6 on AERO interface layering over secured tunnels

o Removed extraneous text in Section 3.7

o Added new detail to the forwarding algorithm in Section 3.9

o Clarified use of fragmentation

o Route optimization now supported for both MNP and non-MNP-based prefixes

o Relays are now seen as link-layer elements in the architecture.

o Built out multicast section in detail.

o New Appendix on implementation considerations for route
   optimization.

Changes from [draft-templin-intarea-6706bis-11](#) to [draft-templin-intrea-6706bis-12](#):

o Introduced Gateways as a new AERO element for connecting
   Correspondent Nodes on INET links

o Introduced terms "Access Network (ANET)" and "Internetwork (INET)"

o Changed "ASP" to "MSP", and "ACP" to "MNP"

o New figure on the relation of Segments to the SPAN and AERO link

o New "S" bit in S/TLLAO to indicate the "Source" S/TLLAO as opposed
   to additional S/TLLAOs

o Changed Interface ID for Servers from 255 to 0xffff

o Significant updates to Route Optimization, NUD, and Mobility
   Management

o New Section on Multicast

o New Section on AERO Clients in the open Internetwork

o New Section on Operation over multiple AERO links (VLANs over the
   SPAN)

o New Sections on DNS considerations and Transition considerations

o

Changes from [draft-templin-intarea-6706bis-10](#) to [draft-templin-intrea-6706bis-11](#):

o Added The SPAN

Changes from [draft-templin-intarea-6706bis-09](#) to [draft-templin-intrea-6706bis-10](#):

o Orphaned packets in flight (e.g., when a neighbor cache entry is
   in the DEPARTED state) are now forwarded at the link layer instead
   of at the network layer. Forwarding at the network layer can
   result in routing loops and/or excessive delays of forwarded
   packets while the routing system is still reconverging.

o Update route optimization to clarify the unsecured nature of the
   first NS used for route discovery

o Many cleanups and clarifications on ND messaging parameters

Changes from draft-templin-intarea-6706bis-08 to draft-templin-
intrea-6706bis-09:

o Changed PRL to "MAP list"

o For neighbor cache entries, changed "static" to "symmetric", and
   "dynamic" to "asymmetric"

o Specified Proxy RS/RA exchanges with Servers on behalf of Clients

o Added discussion of unsolicited NAs in Section 3.16, and included
   forward reference to Section 3.18

o Added discussion of AERO Clients used as critical infrastructure
   elements to connect fixed networks.

o Added network-based VPN under security considerations

Changes from draft-templin-intarea-6706bis-07 to draft-templin-
intrea-6706bis-08:

o New section on AERO-Aware Access Router

Changes from draft-templin-intarea-6706bis-06 to draft-templin-
intrea-6706bis-07:

o Added "R" bit for release of PDs. Now have a full RS/RA service
   that can do PD without requiring DHCPv6 messaging over-the-air

o Clarifications on solicited vs unsolicited NAs

o Clarified use of MAX_NEIGHBOR_ADVERTISEMENTS for the purpose of
   increase reliability

Changes from draft-templin-intarea-6706bis-05 to draft-templin-
intrea-6706bis-06:

o Major re-work and simplification of Route Optimization function

o Added Distributed Mobility Management (DMM) and Mobility Anchor
   Point (MAP) terminology

o New section on "AERO Critical Infrastructure Element
   Considerations" demonstrating low overall cost for the service

o minor text revisions and deletions

o removed extraneous appendices

Changes from draft-templin-intarea-6706bis-04 to draft-templin-
intrea-6706bis-05:

o New Appendix E on S/TLLAO Extensions for special-purpose links.
   Discussed ATN/IPS as example.

o New sentence in introduction to declare appendices as non-
   normative.

Changes from draft-templin-intarea-6706bis-03 to draft-templin-
intrea-6706bis-04:

o Added definitions for Potential Router List (PRL) and secure
   enclave

o Included text on mapping transport layer port numbers to network
   layer DSCP values

o Added reference to DTLS and DMM Distributed Mobility Anchoring
   working group document

o Reworked Security Considerations

o Updated references.

Changes from draft-templin-intarea-6706bis-02 to draft-templin-
intrea-6706bis-03:

o Added new section on SEND.

o Clarifications on "AERO Address" section.

o Updated references and added new reference for RFC8086.

o Security considerations updates.

o General text clarifications and cleanup.

Changes from draft-templin-intarea-6706bis-01 to draft-templin-
intrea-6706bis-02:

o Note on encapsulation avoidance in Section 4.

Changes from draft-templin-intarea-6706bis-00 to draft-templin-
intrea-6706bis-01:

o Remove DHCPv6 Server Release procedures that leveraged the old way
   Relays used to "route" between Server link-local addresses

o Remove all text relating to Relays needing to do any AERO-specific
   operations

o Proxy sends RS and receives RA from Server using SEND. Use CGAs
   as source addresses, and destination address of RA reply is to the
   AERO address corresponding to the Client's ACP.

o Proxy uses SEND to protect RS and authenticate RA (Client does not
   use SEND, but rather relies on subnetwork security. When the
   Proxy receives an RS from the Client, it creates a new RS using
   its own addresses as the source and uses SEND with CGAs to send a
   new RS to the Server.

o Emphasize distributed mobility management

o AERO address-based RS injection of ACP into underlying routing
   system.

Changes from draft-templin-aerolink-82 to draft-templin-intarea-
6706bis-00:

o Document use of NUD (NS/NA) for reliable link-layer address
   updates as an alternative to unreliable unsolicited NA.
   Consistent with Section 7.2.6 of RFC4861.

o Server adds additional layer of encapsulation between outer and
   inner headers of NS/NA messages for transmission through Relays
   that act as vanilla IPv6 routers. The messages include the AERO
   Server Subnet Router Anycast address as the source and the Subnet
   Router Anycast address corresponding to the Client's ACP as the
   destination.

o Clients use Subnet Router Anycast address as the encapsulation
   source address when the access network does not provide a
   topologically-fixed address.

Author's Address

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

    Email: fltemplin@acm.org