                 Asymmetric Extended Route Optimization (AERO)
                         draft-templin-aerolink-78.txt

Abstract

   This document specifies the operation of IP over tunnel virtual links
   using Asymmetric Extended Route Optimization (AERO). Nodes attached
   to AERO links can exchange packets via trusted intermediate routers
   that provide forwarding services to reach off-link destinations and
   route optimization services for improved performance. AERO provides
   an IPv6 link-local address format that supports operation of the IPv6
   Neighbor Discovery (ND) protocol and links IPv6 ND to IP forwarding.
   Mobility management, quality of service (QoS) signaling and route
   optimization are naturally supported through dynamic neighbor cache
   updates, while prefix delegation (PD) is supported by the Dynamic
   Host Configuration Protocol for IPv6 (DHCPv6). AERO is a widely-
   applicable tunneling solution especially well suited to mobile
   Virtual Private Networks (VPNs) and other applications as described
   in this document.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   This document specifies the operation of IP over tunnel virtual links
   using Asymmetric Extended Route Optimization (AERO). The AERO link
   can be used for tunneling between neighboring nodes over either IPv6
   or IPv4 networks, i.e., AERO views the IPv6 and IPv4 networks as

equivalent links for tunneling. Nodes attached to AERO links can
exchange packets via trusted intermediate routers that provide
forwarding services to reach off-link destinations and route
optimization services for improved performance [RFC5522].

AERO provides an IPv6 link-local address format that supports
operation of the IPv6 Neighbor Discovery (ND) [RFC4861] protocol and
links IPv6 ND to IP forwarding. Mobility management, quality of
service (QoS) signaling and route optimization are naturally
supported through dynamic neighbor cache updates, while prefix
delegation (PD) is supported by the Dynamic Host Configuration
Protocol for IPv6 (DHCPv6) [RFC3315].

A node's AERO interface can be configured over multiple underlying
interfaces. From the standpoint of IPv6 ND, AERO interface neighbors
therefore may appear to have multiple link-layer addresses (i.e., the
addresses assigned to underlying interfaces). Each link-layer
address is subject to change due to mobility, and link-layer address
changes are signaled by IPv6 ND messaging the same as for any IPv6
link.

AERO is applicable to a wide variety of 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]. AERO is also applicable to
aviation applications for both manned and unmanned aircraft where the
aircraft is treated as a mobile node that can connect an Internet of
Things (IoT). Other applicable use cases are also in scope.

The remainder of this document presents the AERO specification.

2. Terminology

The terminology in the normative references applies; the following
terms are defined within the scope of this document:

AERO link
   a Non-Broadcast, Multiple Access (NBMA) tunnel virtual overlay
   configured over a node's attached IPv6 and/or IPv4 networks. All
   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 network hops. The AERO mechanisms
   can also operate over native link types (e.g., Ethernet, WiFi
   etc.) when a tunnel virtual overlay is not needed.

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 constructed as specified in
   Section 3.4.

AERO node
   a node that is connected to an AERO link.

AERO Client ("Client")
   a node that requests IP Prefix Delegations (PDs) from one or more
   AERO Servers. Following PD, the Client assigns an AERO address to
   the AERO interface for use in IPv6 ND exchanges with other AERO
   nodes. A node that acts as an AERO Client on one AERO interface
   can also act as an AERO Server on a different AERO interface.

AERO Server ("Server")
   a node that configures an AERO interface to provide default
   forwarding services for AERO Clients. The Server assigns an
   administratively provisioned IPv6 link-local unicast address to
   the AERO interface to support the operation of DHCPv6 and the IPv6
   ND protocol. An AERO Server can also act as an AERO Relay.

AERO Relay ("Relay")
   a node that configures an AERO interface to relay IP packets
   between nodes on the same AERO link and/or forward IP packets
   between the AERO link and the native Internetwork. The Relay
   assigns an administratively provisioned IPv6 link-local unicast
   address to the AERO interface the same as for a Server. An AERO
   Relay can also act as an AERO Server.

AERO Proxy ("Proxy")
   a node that provides proxying services for Clients that cannot
   associate directly with Servers, e.g., when the Client is located
   in a secured internal enclave and the Server is located in the
   external Internetwork. The AERO Proxy is a conduit between the
   secured enclave and the outside world in the same manner as for
   common web proxies, and behaves in a similar fashion as for IPv6
   Neighbor Discovery proxies [RFC4389].

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.

underlying network
   a connected IPv6 or IPv4 network routing region over which the
   tunnel virtual overlay is configured.

underlying interface
   an AERO node's interface point of attachment to an underlying
   network.

link-layer address
   an IP address assigned to an AERO node's underlying interface.
   When UDP encapsulation is used, the UDP port number is also
   considered as part of the link-layer address. Link-layer
   addresses are used as encapsulation header source and destination
   addresses.

network layer address
   the source or destination address of the encapsulated IP packet.

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

AERO Service Prefix (ASP)
   an IP prefix associated with the AERO link and from which more-
   specific AERO Client Prefixes (ACPs) are derived.

AERO Client Prefix (ACP)
   an IP prefix derived from an ASP and delegated to a Client, where
   the ACP prefix length must be no shorter than the ASP prefix
   length and must be no longer than 64 for IPv6 or 32 for IPv4.

base AERO address
   the lowest-numbered AERO address from the first ACP delegated to
   the Client (see Section 3.4).

Throughout the document, the simple terms "Client", "Server", "Relay"
and "Proxy" refer to "AERO Client", "AERO Server", "AERO Relay" and
"AERO Proxy", respectively. Capitalization is used to distinguish
these terms from DHCPv6 client/server/relay [RFC3315].

The terminology of DHCPv6 [RFC3315] and IPv6 ND [RFC4861] (including
the names of node variables and protocol constants) applies to this
document. Also throughout the document, the term "IP" is used to
generically refer to either Internet Protocol version (i.e., IPv4 or
IPv6).

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

```
                        .-(::::::::)
                     .-(:::: IP ::::)-.
                     (:: Internetwork ::)
                      `-(::::::::::::)-'
                         `-(::::::)-'
                              |
     +--------------+    +--------+-------+    +--------------+
     |AERO Server S1|    | AERO Relay R1  |    |AERO Server S2|
     |  Nbr: C1, R1 |    |   Nbr: S1, S2  |    |  Nbr: C2, R1 |
     |  default->R1 |    |(X1->S1; X2->S2)|    |  default->R1 |
     |    X1->C1    |    |     ASP A1     |    |    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  |
     |    ACP X1    |             |             |    ACP X2    |
     +------+-------+    .--------+------.      +-----+--------+
            |           (- Proxied Clients -)        |
          .-.            `--------------'          .-.
        ,-(  _)-.                                ,-(  _)-.
       .-(_  IP   )-.   +-------+    +-------+   .-(_  IP   )-.
      (__    EUN      )--|Host H1|    |Host H2|--(__    EUN      )
        `-(_____)-'    +-------+    +-------+    `-(_____)-'
```
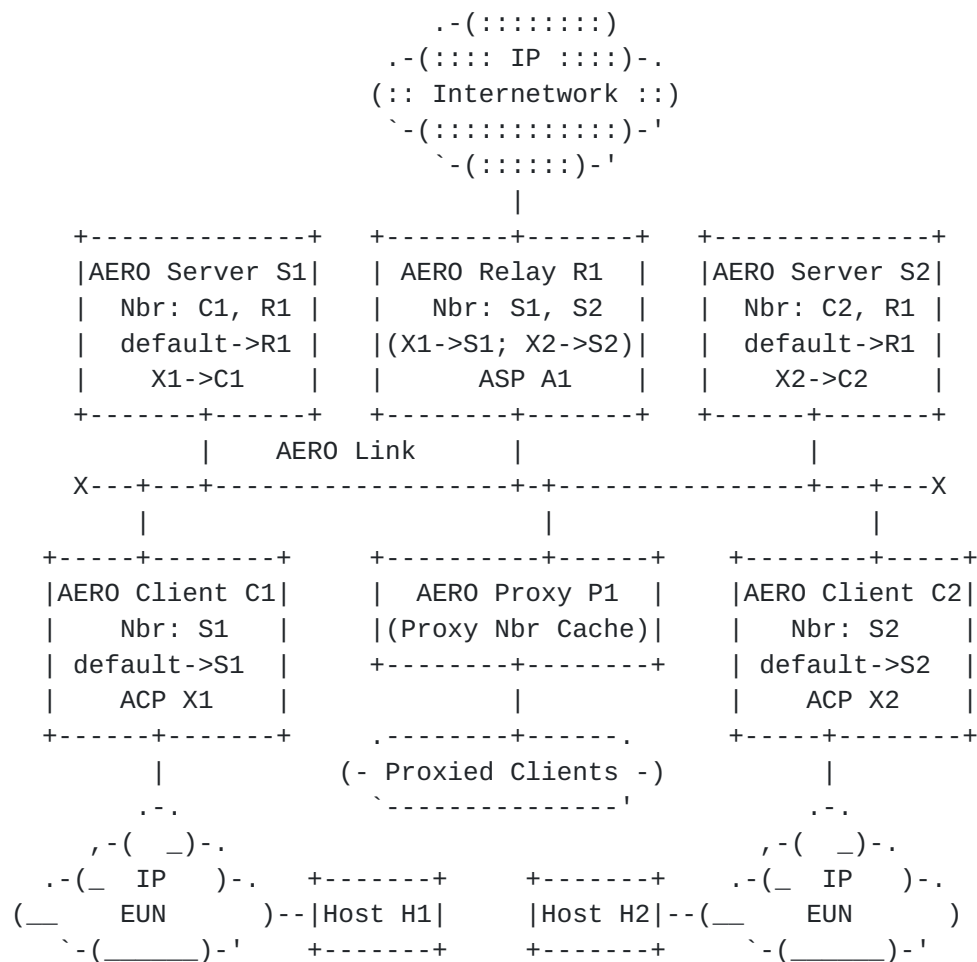
Figure 1: AERO Link Reference Model

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

o AERO Relay R1 aggregates AERO Service Prefix (ASP) A1, acts as a
   default router for its associated Servers (S1 and S2), and
   connects the AERO link to the rest of the IP Internetwork.

o AERO Servers S1 and S2 associate with Relay R1 and also act as
   default routers for their associated Clients C1 and C2.

o AERO Clients C1 and C2 associate with Servers S1 and S2,
   respectively. They receive AERO Client Prefix (ACP) 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 provides proxy services for AERO Clients in secured
   enclaves that cannot associate directly with neighbors in the
   outside world.

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 may be many additional Relays, Servers, Clients and Proxies.

3.2. AERO Node Types

AERO Relays provide default forwarding services to AERO Servers.
Each Relay also peers with Servers and other Relays in a dynamic
routing protocol instance to discover the list of active ACPs (see
Section 3.3). Relays forward packets between neighbors connected to
the same AERO link and also forward packets between the AERO link and
the native IP Internetwork. Relays present the AERO link to the
native Internetwork as a set of one or more AERO Service Prefixes
(ASPs) and serve as a gateway between the AERO link and the
Internetwork. Relays maintain AERO interface neighbor cache entries
for Servers, and maintain an IP forwarding table entry for each AERO
Client Prefix (ACP). AERO Relays can also be configured to act as
AERO Servers.

AERO Servers provide default forwarding services to AERO Clients.
Each Server also peers with Relays in a dynamic routing protocol
instance to advertise its list of associated ACPs (see Section 3.3).
Servers configure a DHCPv6 server function and act as delegating
routers to facilitate Prefix Delegation (PD) exchanges with Clients.
Each delegated prefix becomes an ACP taken from an ASP. Servers
forward packets between AERO interface neighbors, and maintain AERO
interface neighbor cache entries for Relays. They also maintain both
neighbor cache entries and IP forwarding table entries for each of

their associated Clients. AERO Servers can also be configured to act
as AERO Relays.

AERO Clients act as requesting routers to receive ACPs through DHCPv6
PD exchanges with AERO Servers over the AERO link. Each Client can
associate with a single Server or with multiple Servers, e.g., for
fault tolerance, load balancing, etc. Each IPv6 Client receives at
least a /64 IPv6 ACP, and may receive even shorter prefixes.
Similarly, each IPv4 Client receives at least a /32 IPv4 ACP (i.e., a
singleton IPv4 address), and may receive even shorter prefixes.
Clients maintain an AERO interface neighbor cache entry for each of
their associated Servers as well as for each of their correspondent
Clients.

AERO Proxies provide a conduit for AERO Clients located in secured
enclaves to assocaite with Servers located in the outside
Internetwork. The Proxy can either be explicit or transparent. In
the explicit case, the Client sends all of its control plane messages
addressed to the Server to the link-layer address of the Proxy. In
the transparent case, the Client sends all of its control plane
messages to the Server's link-layer address and the Proxy intercepts
them before they leave the secured enclave. In both cases, the Proxy
forwards the Client's control plane messages and data plane messages
to and from the Client's current Server(s). The Proxy may also
discover a more direct route toward a target destination via AERO
route optimization, in which case future outbound data packets would
be forwarded via the more direct route. The Proxy function is
specified in [Section 4](#).

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 the native IP Internetwork interior routing system.
Relays advertise only a small and unchanging set of ASPs to the
native routing system instead of the full dynamically changing set of
ACPs.

In a reference deployment, each AERO 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. All Relays are members of the same
hub AS using a common ASN, and use iBGP to maintain a consistent view
of all active ACPs currently in service.

Each Server maintains a working set of associated ACPs, and
dynamically announces new ACPs and withdraws departed ACPs in its
eBGP updates to Relays. 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.

Each Relay configures a black-hole route for each of its ASPs. By
black-holing the ASPs, the Relay will maintain forwarding table
entries only for the ACPs that are currently active, and packets
destined to all other ACPs will correctly incur Destination
Unreachable messages due to the black hole route. Relays do not send
eBGP updates for ACPs to Servers, but instead originate a default
route. In this way, Servers have only partial topology knowledge
(i.e., they know only about the ACPs of their directly associated
Clients) and they forward all other packets to Relays which have full
topology knowledge.

Scaling properties of the AERO routing system are limited by the
number of BGP routes that can be carried by Relays. At the time of
this writing, the global public Internet BGP routing system manages
more than 500K routes with linear growth and no signs of router
resource exhaustion [BGP]. 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 ASPs. In that case, each Server still
peers with one or more Relays, but the Server institutes route
filters so that it only sends BGP updates to the specific set of
Relays that aggregate the ASP. For example, if the ASP for the AERO
link is 2001:db8::/32, a first set of Relays could service the ASP
segment 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 ACPs with no additional overhead for Servers
and Relays (for example, it should be possible to service 1B /64 ACPs
taken from a /34 ASP and even more for shorter prefixes). In this
way, each set of Relays services a specific set of ASPs that they
advertise to the native routing system, and each Server configures
ASP-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.

Note that in an alternate routing arrangement each set of Relays
could advertise an aggregated ASP for the link into the native
routing system even though each Relay services only smaller segments
of the ASP. In that case, a Relay upon receiving a packet with a
destination address covered by the ASP segment of another Relay can
simply tunnel the packet to the other Relay. The tradeoff then is
the penalty for Relay-to-Relay tunneling compared with reduced
routing information in the native routing system.

3.4. AERO Interface Link-local Addresses

AERO interface link-local address types include administratively-
provisioned addresses and AERO addresses.

Administratively-provisioned addresses are allocated from the range
fe80::/96 and assigned to a Server or Relay's AERO interface.
Administratively-provisioned addresses MUST be managed for uniqueness
by the administrative authority for the AERO link. (Note that fe80::
is the IPv6 link-local subnet router anycast address, and
fe80::ffff:ffff is the address used by Clients to bootstrap AERO
address autoconfiguration. These special addresses are therefore not
available for assignment.)

An AERO address is an IPv6 link-local address with an embedded prefix
based on an ACP and associated with a Client's AERO interface. AERO
addresses remain stable as the Client moves between topological
locations, i.e., even if its link-layer addresses change.

For IPv6, 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 ACPs. For example, if the AERO
Client receives the IPv6 ACP:

    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, AERO addresses are based on an IPv4-mapped IPv6 address
   [RFC4291] formed from an IPv4 ACP and with a Prefix Length of 96 plus
   the ACP prefix length. For example, for the IPv4 ACP 192.0.2.32/28
   the IPv4-mapped IPv6 ACP 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

   When the Server delegates ACPs to the Client, both the Server and
   Client use the lowest-numbered AERO address from the first ACP
   delegation as the "base" AERO address. (For example, for the ACP
   2001:db8:1000:2000::/56 the base address is 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. If
   the Client has multiple AERO addresses (i.e., when there are multiple
   ACPs and/or ACPs with short prefix lengths), the Client originates
   IPv6 ND messages using the base AERO address as the source address
   and accepts and responds to IPv6 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 include
   multiple AERO addresses.

3.5. 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 unicast
   IPv6 ND Neighbor Solicitation (NS), Neighbor Advertisement (NA),
   Router Solicitation (RS), Router Advertisement (RA) and Redirect
   messages for neighbor cache management. AERO interfaces use RS/RA
   messages with embedded DHCPv6 elements per

[I-D.templin-6man-dhcpv6-ndopt] for PD management. AERO interfaces
include routing information in IPv6 ND messages to support route
optimization.

AERO interface ND messages include one or more Source/Target Link-
Layer Address Options (S/TLLAOs) formatted as shown in Figure 2:

```
    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   |V|          Reserved           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Interface ID         |       UDP Port Number         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   |                                                               |
   +                          IP 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|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
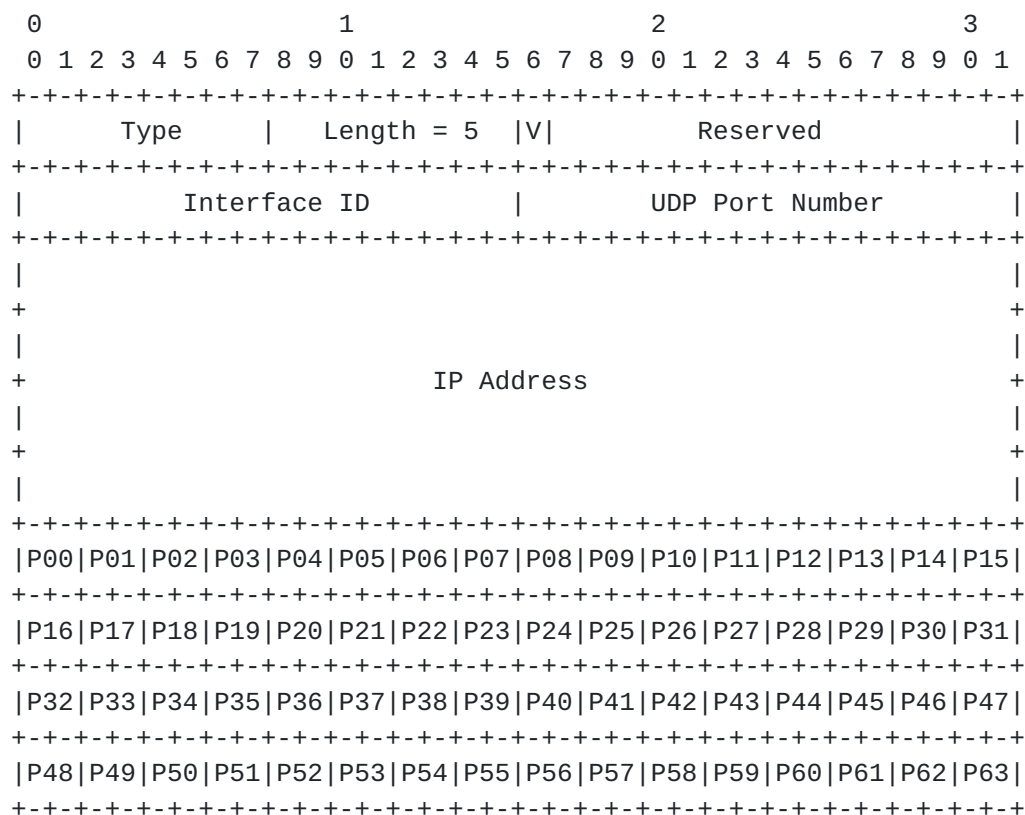
   Figure 2: AERO Source/Target Link-Layer Address Option (S/TLLAO)
                                Format

In this format:

o Type is set to '1' for SLLAO or '2' for TLLAO the same as for IPv6
   ND.

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

o V (Verify) is set to '1' in the first SLLAO in an NS message if
   the recipient is required to forward the NA reply back through the
   reverse chain of Servers and Relays from the target to the source.
   Otherwise, V is set to '0', and is ignored in all other IPv6 ND
   messages.

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

o Interface ID is set to an integer value between 0 and 65535
   corresponding to an underlying interface of the AERO node.

o UDP Port Number and IP Address are set to the addresses used by
   the AERO node when it sends encapsulated packets over the
   underlying interface. When UDP is not used as part of the
   encapsulation, UDP Port Number is set to the value '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 64 Preference values 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.

AERO interfaces may be configured over multiple underlying
interfaces. 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.

If a Client's multiple underlying interfaces are used "one at a time"
(i.e., all other interfaces are in standby mode while one interface
is active), then IPv6 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 underlying interface but with a dynamically
changing link-layer address.

If the Client has multiple active underlying interfaces, then from
the perspective of IPv6 ND it would appear to have multiple link-
layer addresses. In that case, IPv6 ND messages MAY include multiple
S/TLLAOs -- each with an Interface ID that corresponds to a specific
underlying interface of the AERO node.

When an IPv6 ND message includes multiple S/TLLAOs, the first S/TLLAO
MUST correspond to the Client's underlying interface used to transmit
the message.

3.6. AERO Interface Initialization

3.6.1. AERO Relay Behavior

   When a Relay enables an AERO interface, it first assigns an
   administratively-provisioned link-local address fe80::ID to the
   interface. Each fe80::ID address MUST be unique among all AERO nodes
   on the link, and is taken from the range fe80::/96 but excluding the
   special addresses fe80:: and fe80::ffff:ffff. The Relay then engages
   in a dynamic routing protocol session with Servers on the link (see:
   Section 3.3), and advertises its assigned ASPs into the native IP
   Internetwork.

   Each Relay subsequently maintains an IP forwarding table entry for
   each active ACP covered by its ASP(s), and maintains neighbor cache
   entries for Servers on the link. Relays exchange NS/NA messages with
   AERO link neighbors the same as for any AERO node, however they need
   not perform explicit Neighbor Unreachability Detection (NUD) (see:
   Section 3.16) since the dynamic routing protocol already provides
   reachability confirmation.

3.6.2. AERO Server Behavior

   When a Server enables an AERO interface, it assigns an
   administratively-provisioned link-local address fe80::ID the same as
   for Relays. The Server further configures a DHCPv6 server function
   to facilitate DHCPv6 PD exchanges with AERO Clients. The Server
   maintains neighbor cache entries for Relays on the link, and manages
   per-Client neighbor cache entries and IP forwarding table entries
   based on control message exchanges. Each Server also engages in a
   dynamic routing protocol with Relays on the link (see: Section 3.3).

   When the Server receives an NS/RS message from a Client on the AERO
   interface it authenticates the message and returns an NA/RA message.
   The Server further provides a simple link-layer conduit between AERO
   interface neighbors. In particular, when a packet sent by a source
   Client arrives on the Server's AERO interface and is destined to
   another AERO node, the Server forwards the packet from within the
   AERO interface driver at the link layer without ever disturbing the
   network layer.

3.6.3. AERO Client Behavior

   When a Client enables an AERO interface, it uses the special
   administratively-provisioned link-local address fe80::ffff:ffff as
   the source network-layer address in an RS message with an embedded
   DHCPv6 PD Solicit message to obtain one or more ACPs from one or more
   AERO Servers. Each Server processes the message and returns an RA

message with an embedded DHCPv6 PD Reply message with the destination
network-layer address set to the base AERO address. In this way, the
unified RS/RA/DHCPv6 control messages securely perform all
autoconfiguration operations in a single request/response exchange.

After the initial DHCPv6 message exchange, the Client can register
additional links with the Server by sending an RS message over each
link without including a DHCPv6 option. The Server will update its
neighbor cache entry for the Client and return an RA message.

The Client maintains a neighbor cache entry for each of its Servers
and each of its active correspondent Clients. When the Client
receives IPv6 ND messages on the AERO interface it updates or creates
neighbor cache entries, including link-layer address information.

3.6.4. AERO Proxy Behavior

When a Proxy enables an AERO interface, it maintains proxy neighbor
cache entries for Servers on the link, and manages per-Client proxy
neighbor cache entries based on control message exchanges. Proxies
forward packets between their associated Clients and the Client's
associated Servers.

When the Proxy receives an RS message from a Client in the secured
enclave, it forwards the message to a Server selected by the Client
while using its own link-layer address as the source address. When
the Server returns an RA message, the Proxy caches the
autoconfiguration information in the RA and forwards the RA to the
Client while using its own link-layer address as the source address.
The Client, Server and Proxy will then have the necessary state for
managing the proxied neighbor association.

3.7. AERO Interface Neighbor Cache Maintenace

Each AERO interface maintains a conceptual neighbor cache that
includes an entry for each neighbor it communicates with on the AERO
link, the same as for any IPv6 interface [RFC4861]. AERO interface
neighbor cache entires are said to be one of "permanent", "static" or
"dynamic".

Permanent neighbor cache entries are created through explicit
administrative action; they have no timeout values and remain in
place until explicitly deleted. AERO Relays maintain permanent
neighbor cache entries for Servers on the link, and AERO Servers
maintain permanent neighbor cache entries for Relays. Each entry
maintains the mapping between the neighbor's fe80::ID network-layer
address and corresponding link-layer address.

Static neighbor cache entries are created and maintained through
prefix delegation exchanges as specified in Section 3.14, and remain
in place for durations bounded by prefix delegation lifetimes. AERO
Servers and Proxies maintain static neighbor cache entries for each
of their associated Clients, and AERO Proxies and Clients maintain
static neighbor cache entries for each of their associated Servers.

Dynamic neighbor cache entries are created or updated based on
receipt of route optimization messages as specified in Section 3.15,
and are garbage-collected when keepalive timers expire. AERO Clients
and Proxies maintain dynamic neighbor cache entries for each of their
active correspondents with lifetimes based on IPv6 ND messaging
constants.

When an AERO Client receives a valid NS message, it returns an NA
message and also creates or updates a dynamic neighbor cache entry
for the source network-layer and link-layer addresses. The node then
sets an "AcceptTime" variable in the neighbor cache entry to
ACCEPT_TIME seconds and uses this value to determine whether packets
received from the correspondent can be accepted. The node resets
AcceptTime when it receives a new IPv6 ND message, and otherwise
decrements AcceptTime while no IPv6 ND messages have been received.
It is RECOMMENDED that ACCEPT_TIME be set to the default constant
value 40 seconds to allow a 10 second window so that the AERO route
optimization procedure can converge before AcceptTime decrements
below FORWARD_TIME (see below).

When an AERO Client receives a valid NA message that matches its NS
message, it creates or updates a dynamic neighbor cache entry for the
target network-layer and link-layer addresses. The Client then sets
a "ForwardTime" variable in the neighbor cache entry to FORWARD_TIME
seconds and uses this value to determine whether packets can be sent
directly to the correspondent. The node resets ForwardTime when it
receives a new NA, and otherwise decrements ForwardTime while no
further NA messages have been received. It is RECOMMENDED that
FORWARD_TIME be set to the default constant value 30 seconds to match
the default REACHABLE_TIME value specified for IPv6 ND [RFC4861].

The Client also sets a "MaxRetry" variable to MAX_RETRY to limit the
number of keepalives sent when a correspondent may have gone
unreachable. It is RECOMMENDED that MAX_RETRY be set to 3 the same
as described for IPv6 ND address resolution in Section 7.3.3 of
[RFC4861].

Different values for ACCEPT_TIME, FORWARD_TIME and MAX_RETRY MAY be
administratively set, if necessary, to better match the AERO link's
performance characteristics; however, if different values are chosen,
all nodes on the link MUST consistently configure the same values.

Most importantly, ACCEPT_TIME SHOULD be set to a value that is
sufficiently longer than FORWARD_TIME to allow the AERO route
optimization procedure to converge.

When there may be a Network Address Translator (NAT) between the
Client and the Server, or if the path from the Client to the Server
should be tested for reachability, the Client can send periodic RS
messages to the Server without a DHCPv6 option to receive RA replies.
The RS/RA messaging will keep NAT state alive and test Server
reachability without disturbing the DHCPv6 server.

3.8. 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 the AERO tunnel virtual link).
Packets that enter the AERO interface from the network layer are
encapsulated and forwarded into the AERO link, i.e., they are
tunnelled 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 interfaces and/or
neighbor cache entries for neighbors with multiple Interface ID
registrations (see Section 3.5). The AERO node uses each packet's
DSCP value to select an outgoing underlying interface based on the
node's own preference values, and also to select a destination link-
layer address based on the neighbor's underlying interface with the
highest preference value. If multiple outgoing interfaces and/or
neighbor interfaces have a preference of "high", the AERO node sends
one copy of the packet via each of the (outgoing / neighbor)
interface pairs; otherwise, the node sends a single copy of the
packet.

The following sections discuss the AERO interface forwarding
algorithms for Clients, Proxies, Servers and Relays. In the
following discussion, a packet's destination address is said to
"match" if it is a non-link-local address with a prefix covered by an
ASP/ACP, or if it is an AERO address that embeds an ACP, or if it is
the same as an administratively-provisioned link-local address.

3.8.1. Client Forwarding Algorithm

   When an IP packet enters a Client's AERO interface from the network
   layer the Client searches for a dynamic neighbor cache entry that
   matches the destination. If there is a match, the Client uses one or
   more link-layer addresses in the entry as the link-layer addresses
   for encapsulation and admits the packet into the AERO link.
   Otherwise, the Client uses the link-layer address in a static
   neighbor cache entry for a Server as the encapsulation address
   (noting that the "Server" may actually be a Proxy on the path to the
   real Server).

   When an IP packet enters a Client's AERO interface from the link-
   layer, if the destination matches one of the Client's ACPs or link-
   local addresses the Client decapsulates the packet and delivers it to
   the network layer. Otherwise, the Client drops the packet silently.

3.8.2. Proxy Forwarding Algorithm

   When the Proxy receives a packet from a Client within the secured
   enclave, the Proxy searches for a neighbor cache entry that matches
   the destination. If there is a match, the Proxy uses the link-layer
   address in the neighbor cache entry as the link-layer address for
   encapsulation and admits the packet into the AERO link. Otherwise,
   the Proxy uses the link-layer address in a static neighbor cache
   entry for one of the Client's Servers as the encapsulation address.

   When the Proxy receives a packet from a Server, it searches for a
   neighbor cache entry for a local Client within the secured enclave
   that matches the destination. If there is a match, the Proxy
   forwards the packet to the local Client; otherwise, the Proxy drops
   the packet silently.

3.8.3. Server Forwarding Algorithm

   When an IP packet enters a Server's AERO interface from the network
   layer, the Server searches for a neighbor cache entry for a Client or
   Proxy that matches the destination. If there is a match, the Server
   uses one or more link-layer addresses in the entry as the link-layer
   addresses for encapsulation and admits the packet into the AERO link.
   Otherwise, the Server uses the link-layer address in a neighbor cache
   entry for a Relay (selected through longest-prefix match) as the
   link-layer address for encapsulation.

   When an IP packet enters a Server's AERO interface from the link
   layer, the Server processes the packet as follows:

o if the destination matches one of the Server's own addresses the
   Server decapsulates the packet and forwards it to the network
   layer for local delivery.

o else, if the destination matches a neighbor cache entry for a
   Client the Server first determines whether the neighbor is the
   same as the one it received the packet from. If so, the Server
   MUST drop the packet silently to avoid looping; otherwise, the
   Server uses the neighbor's link-layer address(es) as the
   destination for encapsulation and re-admits the packet into the
   AERO link.

o else, the Server uses the link-layer address in a neighbor cache
   entry for a Relay (selected through longest-prefix match) as the
   link-layer address for encapsulation.

3.8.4. Relay Forwarding Algorithm

When an IP packet enters a Relay's AERO interface from the network
layer, the Relay searches its IP forwarding table for an ACP entry
that matches the destination and otherwise searches for a neighbor
cache entry that matches the destination. If there is a match, the
Relay uses the link-layer address in the corresponding neighbor cache
entry as the link-layer address for encapsulation and forwards the
packet into the AERO link. Otherwise, the Relay drops the packet and
(for non-link-local addresses) returns an ICMP Destination
Unreachable message subject to rate limiting (see: Section 3.13).

When an IP packet enters a Relay's AERO interface from the link-
layer, the Relay processes the packet as follows:

o if the destination does not match an ASP, or if the destination
   matches one of the Relay's own addresses, the Relay decapsulates
   the packet and forwards it to the network layer where it will be
   subject to either local delivery or IP forwarding.

o else, if the destination matches an ACP entry in the IP forwarding
   table, or if the destination matches the link-local address in a
   permanent neighbor cache entry, the Relay first determines whether
   the neighbor is the same as the one it received the packet from.
   If so the Relay MUST drop the packet silently to avoid looping;
   otherwise, the Relay uses the neighbor's link-layer address as the
   destination for encapsulation and re-admits the packet into the
   AERO link.

o else, the Relay drops the packet and (for non-link-local
   addresses) returns an ICMP Destination Unreachable message subject
   to rate limiting (see: Section 3.13).

3.9. **AERO Interface Encapsulation and Re-encapsulation**

   AERO interfaces encapsulate IP 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".

   The AERO interface encapsulates packets per the Generic UDP
   Encapsulation (GUE) procedures in
   [I-D.ietf-nvo3-gue][I-D.herbert-gue-fragmentation], or through an
   alternate encapsulation format (see: Appendix A). 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 IP header. For packets
   undergoing re-encapsulation, the AERO interface instead copies these
   values from the original encapsulation IP 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 3.12.

   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
   encapsulated 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.

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. Decapsulation is per the
   procedures specified for the appropriate encapsulation format.

3.11. AERO Interface Data Origin Authentication

   AERO nodes employ simple data origin authentication procedures for
   encapsulated packets they receive from other nodes on the AERO link.
   In particular:

   o AERO Relays and Servers accept encapsulated packets with a link-
      layer source address that matches a permanent neighbor cache
      entry.

   o AERO Servers accept authentic encapsulated DHCPv6 and IPv6 ND
      messages from Clients, and create or update a static neighbor
      cache entry for the Client based on the specific message type.

   o AERO Clients, Proxies and Servers accept encapsulated packets if
      there is a static neighbor cache entry with a link-layer address
      that matches the packet's link-layer source address.

   o AERO Clients, Proxies and Servers accept encapsulated packets if
      there is a dynamic neighbor cache entry with an AERO address that
      matches the packet's network-layer source address, with a link-
      layer address that matches the packet's link-layer source address,
      and with a non-zero AcceptTime.

   Note that this simple data origin authentication is effective in
   environments in which link-layer addresses cannot be spoofed. In
   other environments, each AERO message must include a signature that
   the recipient can use to authenticate the message origin, e.g., as
   for common VPN systems such as OpenVPN [OVPN]. In environments where
   end systems use end-to-end security, however, it may be sufficient to
   require signatures only for AERO DHCPv6, IPv6 ND and ICMP control
   plane messages and omit signatures for data plane messages.

3.12. AERO Interface Packet Size Issues

   The AERO interface is the node's attachment to the AERO link. The
   AERO interface acts as a tunnel ingress when it sends a packet to an
   AERO link neighbor and as a tunnel egress when it receives a packet
   from an AERO link 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][RFC1981]. However, PTB messages may be
   crafted for malicious purposes such as denial of service, 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 a packet that incurs source fragmentation may
also incur network fragmentation.

IPv6 specifies a minimum link Maximum Transmission Unit (MTU) of 1280
bytes [RFC2460]. 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
[RFC2460], while the minimum MRU for IPv4 is only 576 bytes [RFC1122]
(note that common IPv6 over IPv4 tunnels 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.

All AERO nodes MUST configure the same MTU/MSU values 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.

The network layer proceeds as follow when it presents an IP packet to
the AERO interface. For each IPv4 packet that is larger than the
AERO interface MTU and with the DF bit 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, the interface
(acting as a tunnel ingress) encapsulates the packet. If the
encapsulated packet is larger than the AERO interface MSU the ingress
source-fragments the encapsulated packet into a minimum number of
non-overlapping fragments where the first fragment is no larger than
the MSU and the remaining fragments are no larger than the first.
The ingress then admits each encapsulated packet or fragment into the
tunnel, and for IPv4 sets the DF bit to 0 in the IP encapsulation
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.

Several factors must be considered when fragmentation is needed. For
AERO links over IPv4, the IP ID field is only 16 bits in length,
meaning that fragmentation at high data rates could result in data
corruption due to reassembly misassociations [RFC6864][RFC4963]. For
AERO links over both IPv4 and IPv6, studies have also shown that IP
fragments are dropped unconditionally over some network paths [I-
D.taylor-v6ops-fragdrop]. In environments where IP fragmentation
issues could result in operational problems, the ingress SHOULD
employ intermediate-layer source fragmentation (see: [RFC2764] and
[I-D.herbert-gue-fragmentation]) before appending the outer
encapsulation headers to each fragment. Since the encapsulation
fragment header reduces the room available for packet data, but the
original source has no way to control its insertion, the ingress MUST
include the fragment header length in the ENCAPS length even for
packets in which the header is absent.

3.13. 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 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 3.12.)

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 3 (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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
```
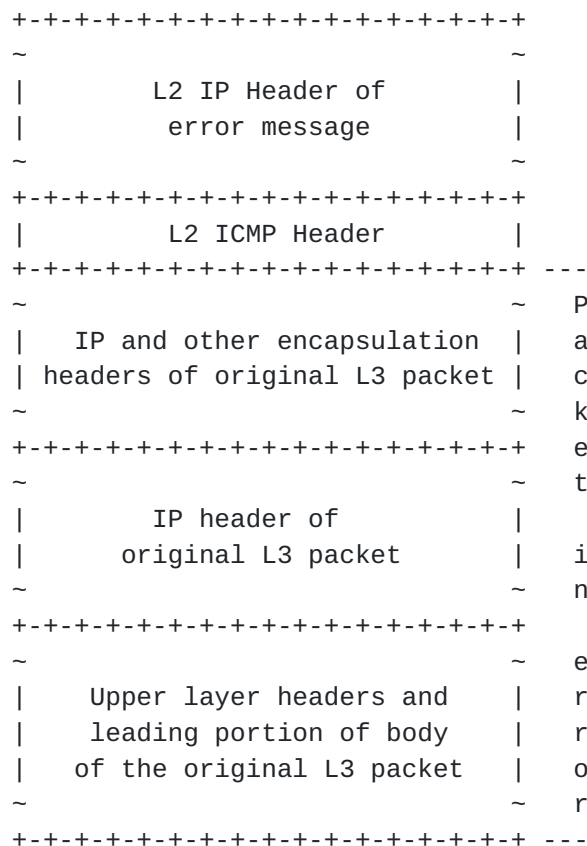
          Figure 3: AERO Interface Link-Layer Error Message Format

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 dynamic neighbor correspondents, the node
   SHOULD test the path to the correspondent using Neighbor
   Unreachability Detection (NUD) (see Section 3.16). If NUD fails
   (or if the node opts to forego NUD) the node SHOULD set
   ForwardTime for the corresponding dynamic 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 static neighbor Servers, the Client SHOULD
   test the path to the Server using NUD. If NUD fails, the Client
   SHOULD associate with a new Server and send a DHCPv6 Release
   message to the old Server as specified in Section 3.17.6.

o When an AERO Server receives persistent link-layer Destination
   Unreachable messages in response to encapsulated packets that it
   sends to one of its static neighbor Clients, the Server SHOULD
   test the path to the Client using NUD. If NUD fails, the Server
   SHOULD cancel the Client's ACP prefix delegation, withdraw its
   route for the ACP from the AERO routing system and delete the
   neighbor cache entry (see Section 3.16 and Section 3.17).

o When an AERO Relay or Server 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, neighbor reachability will be determined by the dynamic
   routing protocol.

When an AERO Relay receives a packet for which the network-layer
destination address is covered by an ASP, 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 first writes the network-layer
source address of the original packet as the destination address of
the message and determines the next hop to the destination. If the
next hop is reached via the AERO interface, the Relay uses the IPv6
address "::" or the IPv4 address "0.0.0.0" as the source address of
the message, then encapsulates the message and forwards it to the
next hop within the AERO interface. Otherwise, the Relay uses one of
its non link-local addresses as the source address of the message and
forwards it via a link outside the AERO interface.

When an AERO node receives an encapsulated packet for which the
reassembly buffer it too small, it drops the packet and returns an
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 of the message and determines the next hop to the
destination. If the next hop is reached via the AERO interface, the
node uses the IPv6 address "::" or the IPv4 address "0.0.0.0" as the
source address of the message, then encapsulates the message and
forwards it to the next hop within the AERO interface. Otherwise,
the node uses one of its non link-local addresses as the source
address of the message and forwards it via a link outside the AERO
interface.

When an AERO node receives any network-layer error message via the
AERO interface, it examines the network-layer destination address.
If the next hop toward the destination is via the AERO interface, the
node re-encapsulates and forwards the message to the next hop within
the AERO interface. Otherwise, if the network-layer source address
is the IPv6 address "::" or the IPv4 address "0.0.0.0", the node
writes one of its non link-local addresses as the source address,
recalculates the IP and/or ICMP checksums then forwards the message
via a link outside the AERO interface.

3.14. AERO Router Discovery, Prefix Delegation and Autoconfiguration

AERO Router Discovery, Prefix Delegation and Autoconfiguration are
coordinated by the DHCPv6 and IPv6 ND control messaging protocols as
discussed in the following Sections.

3.14.1. AERO DHCPv6 and IPv6 ND Service Model

Each AERO Server configures a DHCPv6 server function to facilitate PD
requests from Clients. Each Server is provisioned with a database of
ACP-to-Client ID mappings for all Clients enrolled in the AERO
system, 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. Therefore, no
Server-to-Server DHCPv6 PD state synchronization is necessary, and
Clients can optionally hold separate PDs for the same ACPs from
multiple Servers. In this way, Clients can associate with multiple
Servers, and can receive new PDs from new Servers before deprecating
PDs received from existing Servers. This provides the Client with a
natural fault-tolerance and/or load balancing profile.

AERO Clients and Servers use unicast IPv6 ND messages to maintain
neighbor cache entries the same as for any link. 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.

AERO Clients and Servers include DHCPv6 messages as options in the
RS/RA messages they exchange per [I-D.templin-6man-dhcpv6-ndopt].
The combined DHCPv6/IPv6ND messages can be secured using SEcure
Neighbor Discovery (SEND) [RFC3971].

The following sections specify the Client and Server behavior.

3.14.2. AERO Client Behavior

AERO Clients discover the link-layer addresses of AERO Servers via
static configuration (e.g., from a flat-file map of Server addresses
and locations), or through an automated means such as DNS name
resolution. In the absence of other information, the Client resolves
the FQDN "linkupnetworks.[domainname]" where "linkupnetworks" is a
constant text string and "[domainname]" is a DNS suffix for the
Client's underlying network (e.g., "example.com"). After discovering
the link-layer addresses, the Client associates with one or more of
the corresponding Servers.

To associate with a Server, the Client acts as a requesting router to
request ACPs through a DHCPv6 PD exchange [RFC3315][RFC3633] in
conjunction with standard IPv6 ND Router Discovery. The Client
prepares an RS message with an embedded DHCPv6 Solicit message with
fe80::ffff:ffff as the IPv6 source address, All-Routers multicast as
the IPv6 destination address, the address of the Client's underlying
interface as the link-layer source address and the link-layer address
of the Server as the link-layer destination address.

In the embedded DHCPv6 Solicit message, the Client includes a Client
Identifier option with the Client's DUID, and an Identity Association
for Prefix Delegation (IA_PD) option. If the Client is pre-
provisioned with ACPs associated with the AERO service, it MAY also

include the ACPs in the IA_PD to indicate its preferences to the
DHCPv6 server. The Client finally includes any additional DHCPv6
options.

The Client next includes one or more SLLAOs in the RS formatted as
described in Section 3.5 to register its link-layer address(es) with
the Server. The first SLLAO MUST correspond to the underlying
interface over which the Client will send the RS/DHCPv6 message. The
Client MAY include additional SLLAOs specific to other underlying
interfaces, but if so it MUST have assurance that there will be no
NATs or Proxies on the paths to the Server via those interfaces.
(Otherwise, the Client can register additional link-layer addresses
with the Server by sending subsequent NS/RS messages via different
underlying interfaces after the initial RS/RA exchange). The Server
will use the SLLAOs to populate its link-layer address information
for the Client.

The Client then sends the combined RS/DHCPv6 message to the AERO
Server and waits for an RA reply (see Section 3.14.3) while retrying
MAX_RETRY times until an RA is received. If no RA is received, or if
the RA includes a zero Router Lifetime, the Client SHOULD discontinue
autoconfiguration attempts through this Server and try another
Server. Otherwise, the Client processes the embedded DHCPv6 Reply
message and verifies that the message contains valid ACPs in IA_PD
options.

Next, the Client creates a static neighbor cache entry with the
Server's link-local address as the network-layer address and the
Server's encapsulation source address as the link-layer address. The
Client then autoconfigures AERO addresses for each of the delegated
ACPs and assigns the base AERO addresses to the AERO interface.

The Client next examines the P bit in the RA message flags field
[RFC5175]. If the P bit value was 1, the Client can infer that there
is a NAT or Proxy on the path to the Server. The Client also caches
any ASPs included in Route Information Options (RIOs) [RFC4191] as
ASPs to associate with the AERO link, and assigns the MTU/MSU values
in the MTU options to its AERO interface while configuring an
appropriate MRU. This configuration information applies to the AERO
link as a whole, and all AERO nodes will use the same values.

Following autoconfiguration, the Client sub-delegates the ACPs to its
attached EUNs and/or the Client's own internal virtual interfaces as
described in [I-D.templin-v6ops-pdhost]. The Client subsequently
renews its ACP delegations through each of its Servers by sending RS/
DHCPv6 Renew messages.

After the Client registers its Interface IDs and their associated
UDP/IP addresses and 'P(i)' values, it may wish to change one or more
Interface ID registrations, e.g., if an underlying interface becomes
unavailable, if QoS profiles change, etc. To do so, the Client
prepares an unsolicited NA message to send over any available
underlying interface. The NA MUST include a TLLAO specific to the
selected available underlying interface as the first TLLAO and MAY
include any additional TLLAOs specific to other underlying
interfaces. The Client includes fresh 'P(i)' values in each TLLAO to
update the Server's neighbor cache entry. If the Client wishes to
disable some or all DSCPs for an underlying interface, it includes a
TLLAO with 'P(i)' values set to 0 ("disabled").

If the Client wishes to discontinue use of a Server it issues an RS
with an embedded DHCPv6 Release message. When the Server processes
the message, it releases the DHCPv6 PD, deletes its neighbor cache
entry for the Client, withdraws the IP route from the routing system
and returns an RA response with Router Lifetime set to 0.

3.14.3. AERO Server Behavior

AERO Servers act as IPv6 routers and configure a DHCPv6 server
function on their AERO links. AERO Servers arrange to add their
encapsulation layer IP addresses (i.e., their link-layer 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.

When an AERO Server receives a prospective Client's RS with embedded
DHCPv6 Solicit message on its AERO interface, and the Server is too
busy, it SHOULD return an RA with Router Lifetime set to 0.
Otherwise, the Server authenticates the message and processes the
embedded DHCPv6 Solicit message. The Server first determines the
correct ACPs to delegate to the Client by searching the Client
database. When the Server delegates the ACPs, it also creates an IP
forwarding table entry for each ACP so that the AERO BGP-based
routing system will propagate the ACPs to the Relays that aggregate
the corresponding ASP (see: Section 3.3).

Next, the Server prepares a DHCPv6 Reply message with IA_PD options
with the delegated ACPs. For IPv4 ACPs, the prefix included in the
IA_PD option is in IPv4-mapped IPv6 address format and with prefix
length set as specified in Section 3.4. The Server then prepares a
unicast RA message using its link-local address (i.e., fe80::ID) as
the network-layer source address, the Client's base AERO address as
the network-layer destination address, the Server's link-layer
address as the source link-layer address, and the source link-layer
address of the RS message as the destination link-layer address. The

Server next sets the P bit in the RA message flags field [RFC5175] to
1 iIf the source link-layer address in the RS message was different
than the address in the first SLLAO to indicate that there is a NAT
or Proxy on the path; otherwise it sets P to 0. The Server then
includes one or more RIOs that encode the ASPs for the AERO link.
The Server also includes two MTU options - the first MTU option
includes the MTU for the link and the second MTU option includes the
MSU for the link (see Section 3.12). The Server finally embeds the
body of the DHCPv6 Reply message in the RA, then sends the RA to the
Client.

The Server next creates a static 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 link-layer address(es) by
recording the information in each SLLAO option indexed by the
Interface ID and including the UDP port number, IP address and P(i)
values. For the first SLLAO in the list, however, the Server records
the actual encapsulation source UDP and IP addresses instead of those
that appear in the SLLAO in case there was a NAT or Proxy in the
path.

After the initial RS/RA/DHCPv6 exchange, the AERO Server maintains
the neighbor cache entry for the Client until the PD lifetimes
expire. If the Client issues a Renew, the Server extends the PD
lifetimes. If the Client issues a Release, or if the Client does not
issue a Renew before the lifetime expires, the Server deletes the
neighbor cache entry for the Client and withdraws the IP routes from
the AERO routing system.

3.14.3.1. Lightweight DHCPv6 Relay Agent (LDRA)

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 IPv6 ND function may be located
in separate modules. In that case, the Server's AERO interface
driver module can act as a Lightweight DHCPv6 Relay Agent
(LDRA)[RFC6221] to relay DHCPv6 messages to and from the DHCPv6
server module.

When the LDRA receives an authentic RS/DHCPv6 message, it extracts
the DHCPv6 message and wraps it in IPv6/UDP headers. 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 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 SLLAO options 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 delivers the DHCPv6
message to be wrapped into 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.

3.15. AERO Interface Route Optimization

When a source Client forwards packets to a prospective correspondent
Client within the same AERO link domain (i.e., one for which the
packet's destination address is covered by an ASP), the source Client
MAY initiate an AERO link route optimization procedure. The
procedure is based on an exchange of IPv6 ND messages using a chain
of AERO Servers and Relays as a trust basis.

Although the Client is responsible for initiating route optimization,
the Server is the policy enforcement point that determines whether
route optimization is permitted. For example, on some AERO links
route optimization would allow traffic to circumvent critical
network-based traffic inspection points. In those cases, the Server
can simply discard any route optimization messages instead of
forwarding them.

The following sections specify the AERO link route optimization
procedure.

3.15.1. Reference Operational Scenario

Figure 4 depicts the AERO link route optimization reference
operational scenario, using IPv6 addressing as the example (while not
shown, a corresponding example for IPv4 addressing can be easily
constructed). The figure shows an AERO Relay ('R1'), two AERO
Servers ('S1', 'S2'), two AERO Clients ('C1', 'C2') and two ordinary
IPv6 hosts ('H1', 'H2'):

```
         +--------------+  +--------------+  +--------------+
         |   Server S1  |  |   Relay R1   |  |   Server S2  |
         +--------------+  +--------------+  +--------------+
             fe80::2            fe80::1           fe80::3
              L2(S1)            L2(R1)            L2(S2)
                |                 |                 |
      X-----+-----+----------------+----------------+----+----X
            |          AERO Link                    |
          L2(C1)                                  L2(C2)
       fe80::2001:db8:0:0                     fe80::2001:db8:1:0
        +--------------+                       +--------------+
        |AERO Client C1|                       |AERO Client C2|
        +--------------+                       +--------------+
        2001:DB8:0::/48                        2001:DB8:1::/48
              |                                      |
             .-.                                    .-.
           ,-( _)-.   2001:db8:0::1     2001:db8:1::1    ,-( _)-.
         .-(_  IP  )-.   +-------+       +-------+    .-(_  IP  )-.
        (__    EUN    )--|Host H1|       |Host H2|--(__    EUN    )
          `-(_____)-'   +-------+       +-------+     `-(_____)-'
```
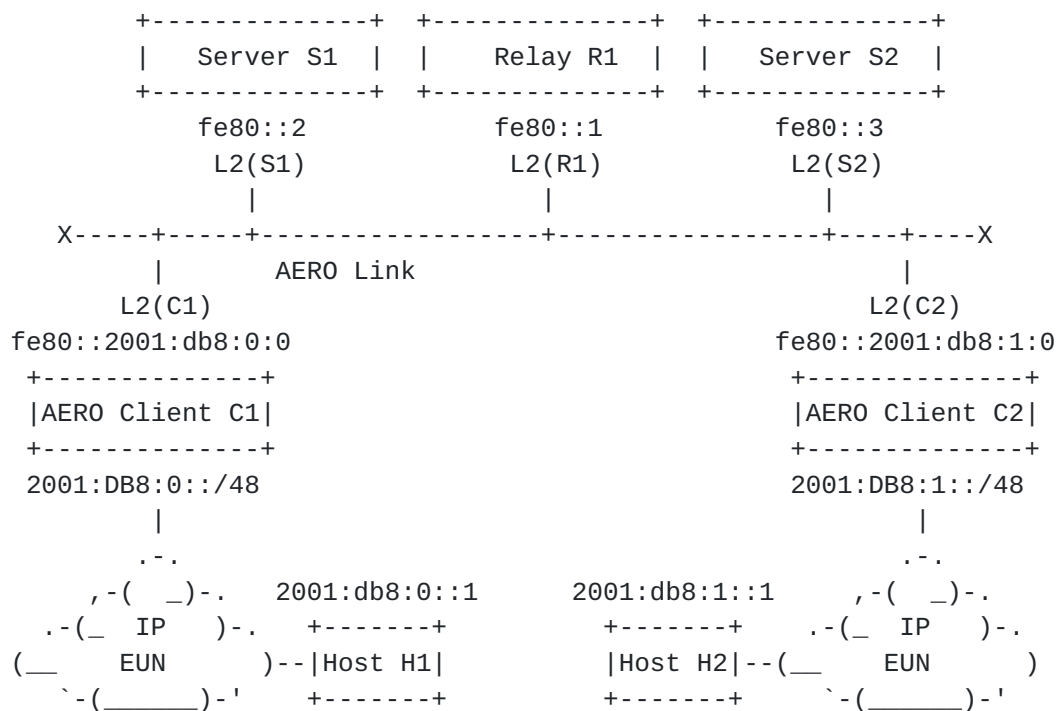
Figure 4: AERO Reference Operational Scenario

In Figure 4, Relay ('R1') assigns the administratively-provisioned
link-local address fe80::1 to its AERO interface with link-layer
address L2(R1), Server ('S1') assigns the address fe80::2 with link-
layer address L2(S1),and Server ('S2') assigns the address fe80::3
with link-layer address L2(S2). Servers ('S1') and ('S2') next
arrange to add their link-layer addresses to a published list of
valid Servers for the AERO link.

AERO Client ('C1') receives the ACP 2001:db8:0::/48 in a DHCPv6 PD
exchange via AERO Server ('S1') then assigns the address
fe80::2001:db8:0:0 to its AERO interface with link-layer address
L2(C1). Client ('C1') configures a default route and neighbor cache
entry via the AERO interface with next-hop address fe80::2 and link-
layer address L2(S1), then sub-delegates the ACP to its attached
EUNs. IPv6 host ('H1') connects to the EUN, and configures the
address 2001:db8:0::1.

AERO Client ('C2') receives the ACP 2001:db8:1::/48 in a DHCPv6 PD
exchange via AERO Server ('S2') then assigns the address
fe80::2001:db8:1:0 to its AERO interface with link-layer address
L2(C2). Client ('C2') configures a default route and neighbor cache
entry via the AERO interface with next-hop address fe80::3 and link-
layer address L2(S2), then sub-delegates the ACP to its attached
EUNs. IPv6 host ('H2') connects to the EUN, and configures the
address 2001:db8:1::1.

3.15.2. **Concept of Operations**

   Again, with reference to Figure 4, when source host ('H1') sends a
   packet to destination host ('H2'), the packet is first forwarded over
   the source host's attached EUN to Client ('C1'). Client ('C1') then
   forwards the packet via its AERO interface to Server ('S1') and also
   sends an NS message toward Client ('C2') via Server ('S1'). If the
   Client is reliant on its trusted Server ('S1') as the primary route
   optimization source, it sets the V bit in the first SLLAO in the NS
   message to '1'.

   Server ('S1') then re-encapsulates and forwards both the packet and
   the NS message out the same AERO interface toward Client ('C2') via
   Relay ('R1'). When Relay ('R1') receives the packet and NS message,
   it consults its forwarding table to discover Server ('S2') as the
   next hop toward Client ('C2'). Relay ('R1') then forwards both the
   packet and the NS message to Server ('S2'), which then forwards them
   to Client ('C2').

   After Client ('C2') receives the NS message, it process the message
   and creates or updates a dynamic neighbor cache entry for Client
   ('C1'). Client ('C2') then examies the V bit in the first SLLAO in
   the NS message. If the V bit is 0 Client ('C2') sends an NA response
   message directly to the link-layer address of Client ('C1') as found
   in the first SLLAO in the NS message; otherwise, Client ('C2') sends
   the NA response to the link-layer address of Server ('S2'). In the
   latter case, when Server ('S2') receives the NA message it re-
   encapsulates the message and forwards it on to Relay ('R1'), which
   forwards the message on to Server ('S1') which forwards the message
   on to Client ('C1').

   After Client ('C1') receives the NA message, it processes the message
   and creates or updates a dynamic neighbor cache entry for Client
   ('C2'). Thereafter, forwarding of packets from Client ('C1') to
   Client ('C2') without involving any intermediate nodes is enabled.
   The mechanisms that support this exchange are specified in the
   following sections.

3.15.3. **Sending NS Messages**

   When a Client forwards a packet with a source address from one of its
   ACPs toward a destination address covered by an ASP (i.e., toward
   another AERO Client connected to the same AERO link), the source
   Client MAY send an NS message forward toward the destination Client
   via the Server.

   In the reference operational scenario, when Client ('C1') forwards a
   packet toward Client ('C2'), it MAY also send an NS message forward

toward Client ('C2'), subject to rate limiting (see Section 8.2 of [RFC4861]). Client ('C1') prepares the NS message as follows:

o the link-layer source address is set to 'L2(C1)' (i.e., the link-layer address of Client ('C1')).

o the link-layer destination address is set to 'L2(S1)' (i.e., the link-layer address of Server ('S1')).

o the network-layer source address is set to fe80::2001:db8:0:0 (i.e., the base AERO address of Client ('C1')).

o the network-layer destination address is set to the AERO address corresponding to the destination address of Client ('C2').

o the Type is set to 135.

o the Target Address is set to the destination address of the packet that triggered route optimization.

o the message includes one or more SLLAOs set to appropriate values for Client ('C1')'s underlying interfaces. The V bit in the first SLLAO is set to 1 if the recipient is required to send the NA reply via the reverse chain of Servers and Relays.

o the message includes one or more RIOs that include Client ('C1')'s ACPs [I-D.templin-6man-rio-redirect].

o the message SHOULD include a Timestamp option and a Nonce option.

Note that the act of sending NS messages is cited as "MAY", since Client ('C1') may have advanced knowledge that the direct path to Client ('C2') would be unusable or otherwise undesirable. If the direct path later becomes unusable after the initial route optimization, Client ('C1') simply allows packets to again flow through Server ('S1').

3.15.4. Re-encapsulating and Relaying the NS

When Server ('S1') receives an NS message from Client ('C1'), it first verifies that the SLLAOs in the NS are a proper subset of the link-layer addresses in Client ('C1')'s neighbor cache entry. If the Client's SLLAOs are not acceptable, Server ('S1') discards the message. Otherwise, Server ('S1') verifies that Client ('C1') is authorized to use the ACPs encoded in the RIOs of the NS. If validation fails, Server ('S1') discards the NS; otherwise, it copies the correct UDP Port number and IP Address for Client ('C1')'s

underlying link into the first SLLAO in case the addresses have been
subject to NAT.

Server ('S1') then examines the network-layer destination address of
the NS to determine the next hop toward Client ('C2') by searching
for the AERO address in the neighbor cache. Since Client ('C2') is
not one of its neighbors, Server ('S1') re-encapsulates the NS and
relays it via Relay ('R1') by changing the link-layer source address
of the message to 'L2(S1)' and changing the link-layer destination
address to 'L2(R1)'. Server ('S1') finally forwards the re-
encapsulated message to Relay ('R1') without decrementing the
network-layer TTL/Hop Limit field.

When Relay ('R1') receives the NS message from Server ('S1') it
determines that Server ('S2') is the next hop toward Client ('C2') by
consulting its forwarding table. Relay ('R1') then re-encapsulates
the NS while changing the link-layer source address to 'L2(R1)' and
changing the link-layer destination address to 'L2(S2)'. Relay
('R1') then relays the NS via Server ('S2').

When Server ('S2') receives the NS message from Relay ('R1') it
determines that Client ('C2') is a neighbor by consulting its
neighbor cache. Server ('S2') then re-encapsulates the NS while
changing the link-layer source address to 'L2(S2)' and changing the
link-layer destination address to 'L2(C2)'. Server ('S2') then
forwards the message to Client ('C2').

3.15.5. Processing NSs and Sending NAs

When Client ('C2') receives the NS message, it accepts the NS only if
the message has a link-layer source address of one of its Servers
(e.g., L2(S2)). Client ('C2') further accepts the message only if it
is willing to serve as a route optimization target.

In the reference operational scenario, when Client ('C2') receives a
valid NS message, it either creates or updates a dynamic neighbor
cache entry that stores the source address of the message as the
network-layer address of Client ('C1') , stores the link-layer
addresses found in the SLLAOs as the link-layer addresses of Client
('C1'), and stores the ACPs encoded in the RIOs of the NS as the ACPs
for Client ('C1'). Client ('C2') then sets AcceptTime for the
neighbor cache entry to ACCEPT_TIME.

After processing the message, Client ('C2') prepares an NA message
response as follows:

o the link-layer source address is set to 'L2(C2)' (i.e., the link-
   layer address of Client ('C2')).

o   the network-layer source address is set to fe80::2001:db8:1:0
    (i.e., the base AERO address of Client ('C2')).

o   the network-layer destination address is set to fe80::2001:db8:0:0
    (i.e., the base AERO address of Client ('C1')).

o   the Type is set to 136.

o   The Target Addrress is set to the Target Address field in the NS
    message.

o   the message includes one or more TLLAOs set to appropriate values
    for Client ('C2')'s underlying interfaces.

o   the message includes one or more RIOs that include Client ('C2')'s
    ACPs [I-D.templin-6man-rio-redirect].

o   the message SHOULD include a Timestamp option and MUST echo the
    Nonce option received in the NS (i.e., if a Nonce option is
    included).

If the V bit in the first SLLAO in the NS message was set to 1,
Client ('C2') then sets the link-layer destination address to
'L2(S2)' (i.e., the link-layer address of Server ('S2')) and sends
the NA message to Server ('S2'). Otherwise, Client ('C2') sets the
link-layer destination address to the address found in the first
SLLAO in the NS message and sends the NA message directly to Client
('C1'), which processes the message according to Section 3.15.7.

3.15.6. Re-encapsulating and Relaying NAs

When Server ('S2') receives an NA message from Client ('C2'), it
first verifies that the TLLAOs in the NA are a proper subset of the
Interface IDs in Client ('C2')'s neighbor cache entry. If the
Client's TLLAOs are not acceptable, Server ('S2') discards the
message. Otherwise, Server ('S2') verifies that Client ('C2') is
authorized to use the ACPs encoded in the RIOs of the NA message. If
validation fails, Server ('S2') discards the NA; otherwise, it copies
the correct UDP Port number and IP Address for Client ('C2')'s
underlying link into the first TLLAO in case the addresses have been
subject to NAT.

Server ('S2') then examines the network-layer destination address of
the NA to determine the next hop toward Client ('C1') by searching
for the AERO address in the neighbor cache. Since Client ('C1') is
not a neighbor, Server ('S2') re-encapsulates the NA and relays it
via Relay ('R1') by changing the link-layer source address of the
message to 'L2(S2)' and changing the link-layer destination address

to 'L2(R1)'. Server ('S2') finally forwards the re-encapsulated
message to Relay ('R1') without decrementing the network-layer TTL/
Hop Limit field.

When Relay ('R1') receives the NA message from Server ('S2') it
determines that Server ('S1') is the next hop toward Client ('C1') by
consulting its forwarding table. Relay ('R1') then re-encapsulates
the NA while changing the link-layer source address to 'L2(R1)' and
changing the link-layer destination address to 'L2(S1)'. Relay
('R1') then relays the NA via Server ('S1').

When Server ('S1') receives the NA message from Relay ('R1') it
determines that Client ('C1') is a neighbor by consulting its
neighbor cache. Server ('S1') then re-encapsulates the NA while
changing the link-layer source address to 'L2(S1)' and changing the
link-layer destination address to 'L2(C1)'. Server ('S1') then
forwards the message to Client ('C1').

3.15.7. Processing NAs

When Client ('C1') receives the NA message, if it had sent the
corresponding NS message with the V bit in the first SLLAO set to 1
it accepts the message only if it has a link-layer source address of
one of its Servers (e.g., 'L2(S1)'). Othewise, Client ('C') accepts
the message unconditionally. Client ('C1') then processes the
message as follows.

In the reference operational scenario, when Client ('C1') receives
the NA message, it either creates or updates a dynamic neighbor cache
entry that stores the source address of the message as the network-
layer address of Client ('C2'), stores the link-layer addresses found
in the TLLAOs as the link-layer addresses of Client ('C2') and stores
the ACPs encoded in the RIOs of the NA as the ACPs for Client ('C2').
Client ('C1') then sets ForwardTime for the neighbor cache entry to
FORWARD_TIME.

Now, Client ('C1') has a neighbor cache entry with a valid
ForwardTime value, while Client ('C2') has a neighbor cache entry
with a valid AcceptTime value. Thereafter, Client ('C1') may forward
ordinary network-layer data packets directly to Client ('C2') without
involving any intermediate nodes, and Client ('C2') can verify that
the packets came from an acceptable source. (In order for Client
('C2') to forward packets to Client ('C1'), a corresponding NS/NA
message exchange is required in the reverse direction; hence, the
mechanism is asymmetric.)

3.15.8. Server-to-Client and Client-to-Server Route Optimization

In some environments, the Server nearest the target Client may need
to serve as a route optimization target, e.g., if direct Client-to-
Client communications are not possible. In that case, when the
source Client sends an NS message the target Server prepares a
corresponding NA the same as if it were the target Client (see:
Section 3.15.5), except that it writes its own link-layer address in
the TLLAO option. The Server must then maintain a dynamic neighbor
cache entry for the source Client.

Similarly, when the source Client must send all packets via its own
Server and cannot act on a route optimization request, the source
Server can send an NS message toward the target Client. The target
Client then prepares a corresponding NA message the same as for
Client-to-Client route optimization and sends the NA message back to
the source Server.

Thereafter, if the target Client moves to a new Server, the old
Server sends unsolicited NA messages with no TLLAOs (subject to rate
limiting) in response to data packets received from a correspondent
node while forwarding the packets themselves to a Relay. The Relay
will then either forward the packets to the new Server if the target
Client has moved, or drop the packets if the target Client is no
longer in the network. The source Client (or Server) then allows
future packets destined to the target Client to again flow through
its own Server (or Relay). Note however that the old Server retains
the neighbor cache entry with its associated AcceptTime since there
may be many packets in flight. When the old Server receives these
packets, it forwards them to a Relay which will forward them to the
departed Client's new Server. AcceptTime will then eventually
decrement to 0 once the correspondent node processes and acts on the
unsolicited NAs.

In any case, a Server MUST NOT send a BGP update to its Relays for
Clients discovered through dynamic route optimization. BGP updates
are only to be sent for the Server's working set of statically-
associated Clients.

3.15.9. Server-to-Server Route Optimization

If neither the source nor target Clients are capable of sending
packets other than via their own Servers, a Server-to-Server route
optimization can still be employed. In that case, the source
Client's Server can send an NS message via a Relay to the AERO
address of the target Client, and the Relay will forward the message
to the target Client's Server. The target Server prepares the NA
message the same as if it were the target Client, except that it

writes its own link-layer address in the TLLAO option then sends the
NA back to the source Server. (The target Server can send the NA
message back to the source Server either directly or via the Relay
according to the security model.)

Thereafter, if the target Client moves to a new Server, the old
Server sends unsolicited NA messages with no TLLAOs (subject to rate
limiting) in response to data packets received from a correspondent
node while forwarding the packets themselves to a Relay. The Relay
will then either forward the packets to the new Server if the target
Client has moved, or drop the packets if the target Client is no
longer in the network. The source Server then allows future packets
destined to the target Client to again flow through a Relay. Note
however that the old Server retains the neighbor cache entry with its
associated AcceptTime since there may be many packets in flight.
When the old Server receives these packets, it forwards them to a
Relay which will forward them to the departed Client's new Server.
AcceptTime will then eventually decrement to 0 once the correspondent
node processes and acts on the unsolicited NAs.

In any case, a Server MUST NOT send a BGP update to its Relays for
correspondents discovered through dynamic route optimization. BGP
updates are only to be sent for the Server's working set of
statically-associated Clients..

3.16. Neighbor Unreachability Detection (NUD)

AERO nodes perform Neighbor Unreachability Detection (NUD) by sending
unicast NS messages with SLLAOs to elicit solicited NA messages from
neighbors the same as described in [RFC4861]. NUD is performed
either reactively in response to persistent L2 errors (see
Section 3.13) or proactively to update neighbor cache entry timers
and/or link-layer address information.

When an AERO node sends an NS/NA message, it uses one of its link-
local addresses as the IPv6 source address and a link-local address
of the neighbor as the IPv6 destination address. When an AERO node
receives an NS message or a solicited NA message, it accepts the
message if it has a neighbor cache entry for the neighbor; otherwise,
it ignores the message.

When route optimization directs a source AERO node to a target AERO
node, the source node SHOULD proactively test the direct path by
sending an initial NS message to elicit a solicited NA response.
While testing the path, the source node can optionally continue
sending packets via its default router, maintain a small queue of
packets until target reachability is confirmed, or (optimistically)
allow packets to flow directly to the target.

While data packets are still flowing, the source node thereafter
periodically tests the direct path to the target node (see
Section 7.3 of [RFC4861]) in order to keep dynamic neighbor cache
entries alive. When the target node receives a valid NS message, it
resets AcceptTime to ACCEPT_TIME and updates its cached link-layer
addresses (if necessary). When the source node receives a solicited
NA message, it resets ForwardTime to FORWARD_TIME and updates its
cached link-layer addresses (if necessary). If the source node is
unable to elicit a solicited NA response from the target node after
MaxRetry attempts, it SHOULD set ForwardTime to 0. Otherwise, the
source node considers the path usable and SHOULD thereafter process
any link-layer errors as a hint that the direct path to the target
node has either failed or has become intermittent.

When ForwardTime for a dynamic neighbor cache entry expires, the
source node resumes sending any subsequent packets via a Server (or
Relay) and may (eventually) attempt to re-initiate the AERO route
optimization process. When AcceptTime for a dynamic neighbor cache
entry expires, the target node discards any subsequent packets
received directly from the source node. When both ForwardTime and
AcceptTime for a dynamic neighbor cache entry expire, the node
deletes the neighbor cache entry.

3.17. Mobility Management and Quality of Service (QoS)

3.17.1. Forwarding Packets on Behalf of Departed Clients

When a Server receives packets with destination addresses that do not
match one of its static neighbor cache Clients, it forwards the
packets packets to a Relay and also returns an unsolicited NA message
to sender with no TLLAOs. The packets will be delivered to the
target Client's new location, and the sender will realize that it
needs to deprecate its routing inforrmation that associated the
target with this Server.

3.17.2. Announcing Link-Layer Address and QoS Changes

When a Client needs to change its link-layer addresses, e.g., due to
a mobility event, it sends unsolicited NAs to its neighbors using the
new link-layer address as the source address and with TLLAOs that
include the new Client UDP Port Numer, IP Address and P(i) values.
(Note that this same mechanism is used to change the Client's QoS
parameters even if the Client's link-layer address itself is
unchanged.)

The Client MAY send up to MaxRetry unsolicited NA messages in
parallel with sending actual data packets in case one or more NAs are

lost. If all NAs are lost, the Client will eventually invoke NUD by
sending NS messages that include SLLAOs.

3.17.3. Bringing New Links Into Service

When a Client needs to bring new underlying interfaces into service
(e.g., when it activates a new data link), it sends unsolicited NAs
to its neighbors using the new link-layer address as the source
address and with TLLAOs that include the new Client link-layer
information.

3.17.4. Removing Existing Links from Service

When a Client needs to remove existing underlying interfaces from
service (e.g., when it de-activates an existing data link), it sends
unsolicited NAs to its neighbors with TLLAOs with UDP Port, IP
Address and all P(i) values set to 0.

If the Client needs to send the unsolicited NAs over an underlying
interface other than the one being removed from service, it MUST
include a current TLLAO for the sending interface as the first TLLAO
and include TLLAOs for any underlying interface being removed from
service as additional TLLAOs.

3.17.5. Implicit Mobility Management

AERO interface neighbors MAY include a configuration knob that allows
them to perform implicit mobility management in which no IPv6 ND
messaging is used. In that case, the Client only transmits packets
over a single interface at a time, and the neighbor always observes
packets arriving from the Client from the same link-layer source
address.

If the Client's underlying interface address changes (either due to a
readdressing of the original interface or switching to a new
interface) the neighbor immediately updates the neighbor cache entry
for the Client and begins accepting and sending packets to the
Client's new link-layer 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.

3.17.6. Moving to a New Server

When a Client associates with a new Server, it performs the Client
procedures specified in Section 3.14.2.

When a Client disassociates with an existing Server, it sends a
DHCPv6 Release message via a new Server with its base AERO address as
the network-layer source address and the unicast link-local address
of the old Server as the network-layer destination address. The new
Server then encapsulates the Release message in a DHCPv6 Relay-
Forward message header, writes the Client's source address in the
'peer-address' field, and writes its own link-local address in the IP
source address (i.e., the new Server acts as a DHCPv6 relay agent).
The new Server then forwards the message to a Relay, which forwards
the message to the old Server based on the network-layer destination
address.

When the old Server receives the Release, it releases the DHCPv6 PDs
and deletes the Client's ACP routes. The old Server then deletes the
Client's neighbor cache entry so that any in-flight packets will be
forwarded via a Relay to the new Server, which will forward them to
the Client. The old Server finally returns a DHCPv6 Relay-Reply
message via a Relay to the new Server, which will decapsulate the
DHCPv6 Reply message and forward it to the Client.

When the new Server forwards the Reply message, the Client can delete
both the default route and the neighbor cache entry for the old
Server. (Note that since Release/Reply messages may be lost in the
network the Client SHOULD retry until it gets a Reply indicating that
the Release was successful. If the Client does not receive a Reply
after MaxRetry attempts, the old Server may have failed and the
Client should discontinue its Release attempts.)

Note that this DHCPv6 relay-chaining approach is provided to avoid
failures, e.g., due to temporary routing fluctuations. In
particular, the Client should always be able to forward messages via
its new Server but may not always be able to send messages directly
to an old Server. But, the new Server and Old Server should always
be able to exchange messages with one another.

Finally, Clients SHOULD NOT move rapidly between Servers in order to
avoid causing excessive oscillations in the AERO routing system.
Such oscillations could result in intermittent reachability for the
Client itself, while causing little harm to the network. 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, etc.

3.17.7. Packet Queueing for Mobility

AERO Clients and Servers should maintain a small queue of packets
they have recently sent to an AERO neighbor, e.g., a 1 second window.
If the AERO neighbor moves, the AERO node MAY retransmit the queued

packets to ensure that they are delivered to the AERO neighbor's new
location.

Note that this may have performance implications for asymmetric
paths. For example, if the AERO neighbor moves from a 50Mbps link to
a 128Kbps link, retransmitting a 1 second window could cause
significant congestion. However, any retransmission bursts will
subside after the 1 second window.

3.18. Multicast Considerations

When the underlying network does not support multicast, AERO Clients
map link-scoped multicast addresses to the link-layer address of a
Server, which acts as a multicast forwarding agent. The AERO Client
also serves as an IGMP/MLD Proxy for its EUNs and/or hosted
applications per [RFC4605] while using the link-layer address of the
Server as the link-layer address for all multicast packets.

When the underlying network supports multicast, AERO nodes use the
multicast address mapping specification found in [RFC2529] for IPv4
underlying networks and use a TBD site-scoped multicast mapping for
IPv6 underlying networks. In that case, border routers must ensure
that the encapsulated site-scoped multicast packets do not leak
outside of the site spanned by the AERO link.

4. The AERO Proxy

In some environments, AERO Clients may be located in secured enclaves
(e.g., a corporate enterprise network) that do not allow direct
communications from the Client to the outside world. In that case,
the secured enclave can emply an AERO Proxy.

The AERO Proxy is located at the secured enclave perimeter and
listens for RS/RA/DHCPv6 messages originating from or destined to
AERO Clients located within the enclave. The Proxy acts on these
control messages as follows:

o when the Proxy receives an RS/DHCPv6 message from a Client within
   the secured enclave, it creates a proxy neighbor cache entry for
   the Client in the INCOMPLETE State and caches the Client's link-
   layer address information. The Proxy then forwards the message to
   the external Server indicated by the destination link-layer
   address in the packet while substituting its own external address
   as the source link-layer address.

o when the Proxy receives an RA/DHCPv6 message from an external
   Server, it matches the RA message with the (incomplete) proxy
   neighbor cache entry. The Proxy then caches the route information

in the message as a mapping from the Client's ACPs to the Client's
address within the secured enclave, and sets the neighbor cache
entry state to REACHABLE. The Proxy then forwards the message to
the Client.

After the initial RS/RA/DHCPv6 handshake is concluded, the Proxy can
send unsolicited NA messages to the Client's chosen Server as-needed
to update the Server's neighbor cache entry for the Client (e.g., to
update the Client's QoS parameters). The Proxy also forwards all
data packets originating from the Client to one of its chosen
Servers. At the same time, for destination addresses that match an
ASP the Proxy sends an NS message via the Server to solicit an NA
message from a Server that is currently serving the target Client (or
from the target Client itself). When the Proxy receives the NA
message, it creates a dynamic neighbor cache entry that associates
the source of the NA message as the next-hop toward the routes
adveritsed in the NA RIOs.

From the perspective of the target, the Proxy that sent the route
optimization NS message will appear as if it is an ordinary AERO
Client. However, the target must deliver the NA message directly to
the Proxy (i.e., instead of relaying through the backward chain of
Relays and Servers) since the backwards chain could deliver the NA to
a different Proxy besides the one that produced the NS. For this
reason, the Proxy must set the V bit in the first SLLAO in the NS
message to 0.

After the NS/NA exchange, the Proxy may receive unsolicited NA
messages without TLLAOs from the target in response to data packets
destined to a departed Client. In that case, the Proxy deletes the
routes associated with the target and allows future packets to flow
through the source Server, and can then re-initialize route
optimization as above.

After the NS/NA exchange, the Proxy may receive link-layer error
messages in response to data packets destined to the target. In that
case, the Proxy processes the link-layer error messages as a hint
that the path to the target may be failing discussed in [Section 3.13](#).

5. Operation on AERO Links with /64 ASPs

IPv6 AERO links typically have ASPs that cover many candidate ACPs of
length /64 or shorter. However, in some cases it may be desirable to
use AERO over links that have only a /64 ASP. 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/DHCPv6 PD message to the Server the same as for ordinary AERO links. The Server responds with an RA/DHCPv6 message that includes one or more /128 prefixes (i.e., singleton addresses) that include the /64 ASP 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 ASP 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 3.15.

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.

Note that, due to the nature of the AERO address format, valid IPv6 ACP lengths are either /64 or shorter, or exactly /128 (i.e., prefix lengths between /65 and /127 cannot be accommodated).

6. Implementation Status

An AERO implementation based on OpenVPN (https://openvpn.net/) was announced on the v6ops mailing list on January 10, 2018. The latest version is available at: http://linkupnetworks.net/aero/AERO-OpenVPN-1.0.tgz.

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 version is available at: http://linkupnetworks.net/aero/aero-3.0.3a.tgz.

7. 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.

8. Security Considerations

AERO link security considerations are the same as for standard IPv6 Neighbor Discovery [RFC4861] except that AERO improves on some aspects. In particular, AERO uses a trust basis between Clients and Servers, where the Clients only engage in the AERO mechanism when it is facilitated by a trust anchor.

NS and NA messages SHOULD include a Timestamp option (see Section 5.3 of [RFC3971]) that other AERO nodes can use to verify the message time of origin. NS and RS messages SHOULD include a Nonce option (see Section 5.3 of [RFC3971]) that recipients echo back in corresponding responses. In cases where spoofing cannot be mitigated through other means, however, all AERO IPv6 ND messages should employ SEND [RFC3971], which also protects the DHCPv6 information embedded in RS/RA message options.

AERO links must be protected against link-layer address spoofing attacks in which an attacker on the link pretends to be a trusted neighbor. Links that provide link-layer securing mechanisms (e.g., IEEE 802.1X WLANs) and links that provide physical security (e.g., enterprise network wired LANs) provide a first line of defense, however AERO nodes SHOULD also use securing services such as SEND for Client authentication and network admission control. Following authenticated Client admission and prefix delgation procedures, AERO nodes MUST ensure that the source of data packets corresponds to the node to which the prefixes were delegated.

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.)

AERO Clients, Servers and Relays on the open Internet are susceptible to the same attack profiles as for any Internet nodes. For this reason, IP security SHOULD be used when AERO is employed over unmanaged/unsecured links using securing mechanisms such as IPsec [RFC4301], IKE [RFC5996] and/or TLS [RFC5246]. In some environments, however, the use of end-to-end security from Clients to correspondent

nodes (i.e., other Clients and/or Internet nodes) could obviate the
need for IP security between AERO Clients, Servers and Relays.

AERO Servers and Relays present targets for traffic amplification 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 Relays and Servers 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 can institute rate limits that protect Clients from receiving
packet floods that could DoS low data rate links.

9. Acknowledgements

Discussions both on IETF lists and in private exchanges helped shape
some of the concepts in this work. Individuals who contributed
insights include Mikael Abrahamsson, Mark Andrews, Fred Baker, Bob
Braden, Stewart Bryant, Brian Carpenter, Wojciech Dec, Ralph Droms,
Adrian Farrel, Sri Gundavelli, Brian Haberman, Joel Halpern, Tom
Herbert, Sascha Hlusiak, Lee Howard, Andre Kostur, Ted Lemon, Andy
Malis, Satoru Matsushima, Tomek Mrugalski, Alexandru Petrescu, Behcet
Saikaya, Joe Touch, Bernie Volz, Ryuji Wakikawa, Lloyd Wood and James
Woodyatt. Members of the IESG also provided valuable input during
their review process that greatly improved the document. Discussions
on the v6ops list in the December 2015 through January 2016 timeframe
further helped clarify AERO multi-addressing capabilities. Special
thanks go to Stewart Bryant, Joel Halpern and Brian Haberman for
their shepherding guidance during the publication of the AERO first
edition.

This work has further been encouraged and supported by Boeing
colleagues including Kyle Bae, M. Wayne Benson, Dave Bernhardt, Cam
Brodie, Balaguruna Chidambaram, Irene Chin, Bruce Cornish, Claudiu
Danilov, Wen Fang, Anthony Gregory, Jeff Holland, Ed King, Gene
MacLean III, Rob Muszkiewicz, Sean O'Sullivan, Kent Shuey, Brian
Skeen, Mike Slane, Carrie Spiker, Brendan Williams, Julie Wulff,
Yueli Yang, Eric Yeh and other members of the BR&T and BIT mobile
networking teams. Wayne Benson is especially acknowledged for
converting the AERO proof-of-concept implementation into production-
ready code for OpenVPN.

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

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

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

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

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

o AERO, First Edition [RFC6706]

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

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

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

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

This work is aligned with the Boeing Research and Technology (BR&T)
autonomous systems networking program.

10. References

10.1. Normative References

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

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

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

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

   [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>.

   [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6
              (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460,
              December 1998, <https://www.rfc-editor.org/info/rfc2460>.

   [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>.

   [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins,
              C., and M. Carney, "Dynamic Host Configuration Protocol
              for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July
              2003, <https://www.rfc-editor.org/info/rfc3315>.

   [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic
              Host Configuration Protocol (DHCP) version 6", RFC 3633,
              DOI 10.17487/RFC3633, December 2003,
              <https://www.rfc-editor.org/info/rfc3633>.

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

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

   [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>.

   [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>.

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

   [RFC6434] Jankiewicz, E., Loughney, J., and T. Narten, "IPv6 Node
              Requirements", RFC 6434, DOI 10.17487/RFC6434, December
              2011, <https://www.rfc-editor.org/info/rfc6434>.

10.2. Informative References

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

   [I-D.herbert-gue-fragmentation]
              Herbert, T. and F. Templin, "Fragmentation option for
              Generic UDP Encapsulation", draft-herbert-gue-
              fragmentation-02 (work in progress), October 2015.

   [I-D.ietf-dhc-sedhcpv6]
              Li, L., Jiang, S., Cui, Y., Jinmei, T., Lemon, T., and D.
              Zhang, "Secure DHCPv6", draft-ietf-dhc-sedhcpv6-21 (work
              in progress), February 2017.

   [I-D.ietf-intarea-tunnels]
              Touch, J. and M. Townsley, "IP Tunnels in the Internet
              Architecture", draft-ietf-intarea-tunnels-07 (work in
              progress), June 2017.

   [I-D.ietf-nvo3-gue]
              Herbert, T., Yong, L., and O. Zia, "Generic UDP
              Encapsulation", draft-ietf-nvo3-gue-05 (work in progress),
              October 2016.

   [I-D.templin-6man-dhcpv6-ndopt]
              Templin, F., "The DHCPv6 Option for IPv6 Neighbor
              Discovery", draft-templin-6man-dhcpv6-ndopt-01 (work in
              progress), January 2018.

   [I-D.templin-6man-rio-redirect]
             Templin, F. and j. woodyatt, "Route Information Options in
             IPv6 Neighbor Discovery", draft-templin-6man-rio-
             redirect-05 (work in progress), October 2017.

   [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 Models", draft-
             templin-v6ops-pdhost-18 (work in progress), December 2017.

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

   [RFC0879] Postel, J., "The TCP Maximum Segment Size and Related
             Topics", RFC 879, DOI 10.17487/RFC0879, November 1983,
             <https://www.rfc-editor.org/info/rfc879>.

   [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>.

   [RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation,
              selection, and registration of an Autonomous System (AS)",
              BCP 6, RFC 1930, DOI 10.17487/RFC1930, March 1996,
              <https://www.rfc-editor.org/info/rfc1930>.

   [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery
              for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August
              1996, <https://www.rfc-editor.org/info/rfc1981>.

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

   [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>.

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

   [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>.

   [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi,
              "DNS Extensions to Support IP Version 6", STD 88,
              RFC 3596, DOI 10.17487/RFC3596, October 2003,
              <https://www.rfc-editor.org/info/rfc3596>.

   [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>.

   [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>.

   [RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-
              Network Tunneling", RFC 4459, DOI 10.17487/RFC4459, April
              2006, <https://www.rfc-editor.org/info/rfc4459>.

   [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>.

   [RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol
              (MOBIKE)", RFC 4555, DOI 10.17487/RFC4555, June 2006,
              <https://www.rfc-editor.org/info/rfc4555>.

   [RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name
              System", RFC 4592, DOI 10.17487/RFC4592, July 2006,
              <https://www.rfc-editor.org/info/rfc4592>.

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

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

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

   [RFC4994] Zeng, S., Volz, B., Kinnear, K., and J. Brzozowski,
              "DHCPv6 Relay Agent Echo Request Option", RFC 4994,
              DOI 10.17487/RFC4994, September 2007,
              <https://www.rfc-editor.org/info/rfc4994>.

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

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

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <https://www.rfc-editor.org/info/rfc5246>.

   [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>.

   [RFC5494] Arkko, J. and C. Pignataro, "IANA Allocation Guidelines
              for the Address Resolution Protocol (ARP)", RFC 5494,
              DOI 10.17487/RFC5494, April 2009,
              <https://www.rfc-editor.org/info/rfc5494>.

   [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>.

   [RFC5720] Templin, F., "Routing and Addressing in Networks with
              Global Enterprise Recursion (RANGER)", RFC 5720,
              DOI 10.17487/RFC5720, February 2010,
              <https://www.rfc-editor.org/info/rfc5720>.

   [RFC5844] Wakikawa, R. and S. Gundavelli, "IPv4 Support for Proxy
              Mobile IPv6", RFC 5844, DOI 10.17487/RFC5844, May 2010,
              <https://www.rfc-editor.org/info/rfc5844>.

   [RFC5949] Yokota, H., Chowdhury, K., Koodli, R., Patil, B., and F.
              Xia, "Fast Handovers for Proxy Mobile IPv6", RFC 5949,
              DOI 10.17487/RFC5949, September 2010,
              <https://www.rfc-editor.org/info/rfc5949>.

   [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen,
              "Internet Key Exchange Protocol Version 2 (IKEv2)",
              RFC 5996, DOI 10.17487/RFC5996, September 2010,
              <https://www.rfc-editor.org/info/rfc5996>.

   [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>.

   [RFC6204] Singh, H., Beebee, W., Donley, C., Stark, B., and O.
              Troan, Ed., "Basic Requirements for IPv6 Customer Edge
              Routers", RFC 6204, DOI 10.17487/RFC6204, April 2011,
              <https://www.rfc-editor.org/info/rfc6204>.

   [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>.

   [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
              and A. Bierman, Ed., "Network Configuration Protocol
              (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
              <https://www.rfc-editor.org/info/rfc6241>.

   [RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility
              Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July
              2011, <https://www.rfc-editor.org/info/rfc6275>.

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

   [RFC6422] Lemon, T. and Q. Wu, "Relay-Supplied DHCP Options",
              RFC 6422, DOI 10.17487/RFC6422, December 2011,
              <https://www.rfc-editor.org/info/rfc6422>.

   [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>.

   [RFC6691] Borman, D., "TCP Options and Maximum Segment Size (MSS)",
              RFC 6691, DOI 10.17487/RFC6691, July 2012,
              <https://www.rfc-editor.org/info/rfc6691>.

   [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>.

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

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

   [RFC6939] Halwasia, G., Bhandari, S., and W. Dec, "Client Link-Layer
              Address Option in DHCPv6", RFC 6939, DOI 10.17487/RFC6939,
              May 2013, <https://www.rfc-editor.org/info/rfc6939>.

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

   [RFC7078] Matsumoto, A., Fujisaki, T., and T. Chown, "Distributing
              Address Selection Policy Using DHCPv6", RFC 7078,
              DOI 10.17487/RFC7078, January 2014,
              <https://www.rfc-editor.org/info/rfc7078>.

   [TUNTAP]   Wikipedia, W., "http://en.wikipedia.org/wiki/TUN/TAP",
              October 2014.

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 5 shows the AERO IP-in-IP encapsulation format before any
   fragmentation is applied:
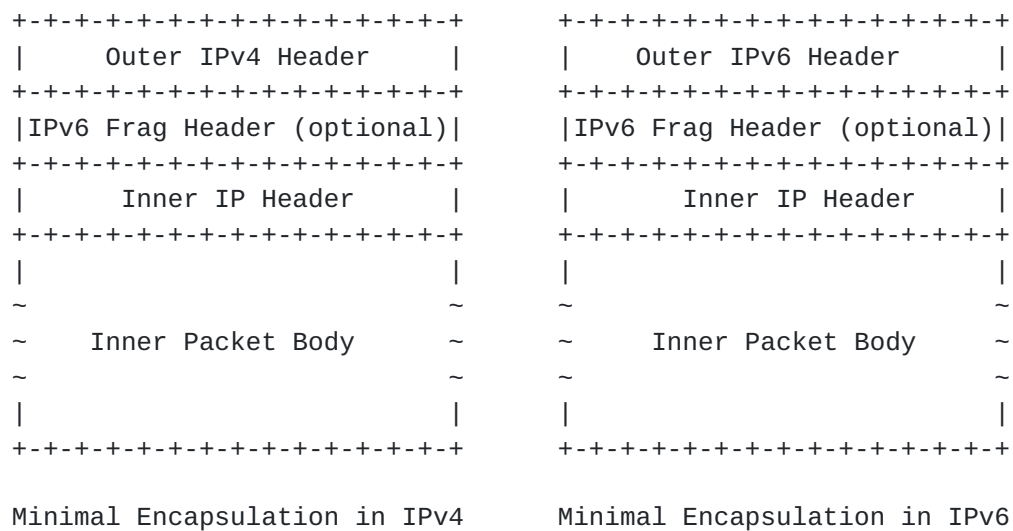
```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+-+-+
    |    Outer IPv4 Header    |        |    Outer IPv6 Header    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+-+-+
    |IPv6 Frag Header (optional)|      |IPv6 Frag Header (optional)|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+-+-+
    |    Inner IP Header      |        |    Inner IP Header      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        |        |                          |
    ~                        ~        ~                          ~
    ~    Inner Packet Body   ~        ~    Inner Packet Body     ~
    ~                        ~        ~                          ~
    |                        |        |                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+-+-+

    Minimal Encapsulation in IPv4     Minimal Encapsulation in IPv6
```

         Figure 5: Minimal Encapsulation Format using IP-in-IP

   Figure 6 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    ~
    ~                            ~
    |                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
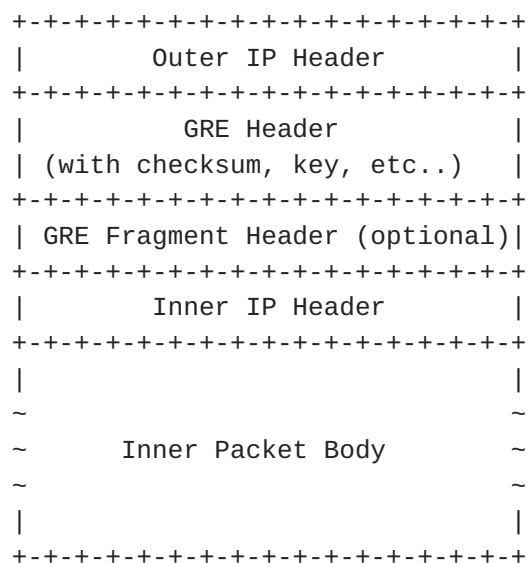
                Figure 6: Minimal Encapsulation Using GRE

   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 and SSL/TLS. 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. When to Insert an Encapsulation Fragment Header

An encapsulation fragment header is inserted when the AERO tunnel ingress needs to apply fragmentation to accommodate packets that must be delivered without loss due to a size restriction. Fragmentation is performed on the inner packet while encapsulating each inner packet fragment in outer IP and encapsulation layer headers that differ only in the fragment header fields.

The fragment header can also be inserted in order to include a coherent Identification value with each packet, e.g., to aid in Duplicate Packet Detection (DPD). In this way, network nodes can cache the Identification values of recently-seen packets and use the cached values to determine whether a newly-arrived packet is in fact a duplicate. The Identification value within each packet could further provide a rough indicator of packet reordering, e.g., in cases when the tunnel egress wishes to discard packets that are grossly out of order.

In some use cases, there may be operational assurance that no fragmentation of any kind will be necessary, or that only occasional large control messages will require fragmentation. In that case, the encapsulation fragment header can be omitted and ordinary fragmentation of the outer IP protocol version can be applied when necessary.

Appendix C. Autoconfiguration for Constrained Platforms

On some platforms (e.g., popular cell phone operating systems), the act of assigning a default IPv6 route and/or assigning an address to an interface may not be permitted from a user application due to

security policy. Typically, those platforms include a TUN/TAP
interface [TUNTAP] that acts as a point-to-point conduit between user
applications and the AERO interface. In that case, the Client can
instead generate a "synthesized RA" message. The message conforms to
[RFC4861] and is prepared as follows:

o the IPv6 source address is the Client's AERO address

o the IPv6 destination address is all-nodes multicast

o the Router Lifetime is set to a time that is no longer than the
   ACP DHCPv6 lifetime

o the message does not include a Source Link Layer Address Option
   (SLLAO)

o the message includes a Prefix Information Option (PIO) with a /64
   prefix taken from the ACP as the prefix for autoconfiguration

The Client then sends the synthesized RA message via the TUN/TAP
interface, where the operating system kernel will interpret it as
though it were generated by an actual router. The operating system
will then install a default route and use StateLess Address
AutoConfiguration (SLAAC) to configure an IPv6 address on the TUN/TAP
interface. Methods for similarly installing an IPv4 default route
and IPv4 address on the TUN/TAP interface are based on synthesized
DHCPv4 messages [RFC2131].

Appendix D. Operational Deployment Alternatives

AERO can be used in many different variations based on the specific
use case. The following sections discuss variations that adhere to
the AERO principles while allowing selective application of AERO
components.

D.1. Operation on AERO Links Without DHCPv6 Services

When Servers on the AERO link do not provide DHCPv6 services,
operation can still be accommodated through administrative
configuration of ACPs on AERO Clients. In that case, administrative
configurations of AERO interface neighbor cache entries on both the
Server and Client are also necessary. However, this may interfere
with the ability for Clients to dynamically change to new Servers,
and can expose the AERO link to misconfigurations unless the
administrative configurations are carefully coordinated.

D.2. Operation on Server-less AERO Links

   In some AERO link scenarios, there may be no Servers on the link and/
   or no need for Clients to use a Server as an intermediary trust
   anchor. In that case, each Client acts as a Server unto itself to
   establish neighbor cache entries by performing direct Client-to-
   Client IPv6 ND message exchanges, and some other form of trust basis
   must be applied so that each Client can verify that the prospective
   neighbor is authorized to use its claimed ACP.

   When there is no Server on the link, Clients must arrange to receive
   ACPs and publish them via a secure alternate PD authority through
   some means outside the scope of this document.

D.3. Operation on Client-less AERO Links

   In some environments, the AERO service may be useful for mobile nodes
   that do not implement the AERO Client function and do not perform
   encapsulation. For example, if the mobile node has a way of
   injecting its ACP into the access subnetwork routing system an AERO
   Server connected to the same access network can accept the ACP prefix
   injection as an indication that a new mobile node has come onto the
   subnetwork. The Server can then inject the ACP into the BGP routing
   system the same as if an AERO Client/Server DHCPv6 PD exchange had
   occurred. If the mobile node subsequently withdraws the ACP from the
   access network routing system, the Server can then withdraw the ACP
   from the BGP routing system.

   In this arrangement, AERO Servers and Relays are used in exactly the
   same ways as for environments where DHCPv6 Client/Server exchanges
   are supported. However, the access subnetwork routing systems must
   be capable of accommodating rapid ACP injections and withdrawals from
   mobile nodes with the understanding that the information must be
   propagated to all routers in the system. Operational experience has
   shown that this kind of routing system "churn" can lead to overall
   instability and routing system inconsistency.

D.4. Manually-Configured AERO Tunnels

   In addition to the dynamic neighbor discovery procedures for AERO
   link neighbors described above, AERO encapsulation can be applied to
   manually-configured tunnels. In that case, the tunnel endpoints use
   an administratively-provisioned link-local address and exchange NS/NA
   messages the same as for dynamically-established tunnels.

D.5. Encapsulation Avoidance on Relay-Server Dedicated Links

In some environments, AERO Servers and Relays may be connected by
dedicated point-to-point links, e.g., high speed fiberoptic leased
lines. In that case, the Servers and Relays can participate in the
AERO link the same as specified above but can avoid encapsulation
over the dedicated links. In that case, however, the links would be
dedicated for AERO and could not be multiplexed for both AERO and
non-AERO communications.

D.6. Encapsulation Protocol Version Considerations

A source Client may connect only to an IPvX underlying network, while
the target Client connects only to an IPvY underlying network. In
that case, the target and source Clients have no means for reaching
each other directly (since they connect to underlying networks of
different IP protocol versions) and so must ignore any route
optimization messages and continue to send packets via their Servers.

D.7. Extending AERO Links Through Security Gateways

When an enterprise mobile node moves from a campus LAN connection to
a public Internet link, it must re-enter the enterprise via a
security gateway that has both a physical interface connection to the
Internet and a physical interface connection to the enterprise
internetwork. This most often entails the establishment of a Virtual
Private Network (VPN) link over the public Internet from the mobile
node to the security gateway. During this process, the mobile node
supplies the security gateway with its public Internet address as the
link-layer address for the VPN. The mobile node then acts as an AERO
Client to negotiate with the security gateway to obtain its ACP.

In order to satisfy this need, the security gateway also operates as
an AERO Server with support for AERO Client proxying. In particular,
when a mobile node (i.e., the Client) connects via the security
gateway (i.e., the Server), the Server provides the Client with an
ACP in a DHCPv6 PD exchange the same as if it were attached to an
enterprise campus access link. The Server then replaces the Client's
link-layer source address with the Server's enterprise-facing link-
layer address in all AERO messages the Client sends toward neighbors
on the AERO link. The AERO messages are then delivered to other
nodes on the AERO link as if they were originated by the security
gateway instead of by the AERO Client. In the reverse direction, the
AERO messages sourced by nodes within the enterprise network can be
forwarded to the security gateway, which then replaces the link-layer
destination address with the Client's link-layer address and replaces
the link-layer source address with its own (Internet-facing) link-
layer address.

   After receiving the ACP, the Client can send IP packets that use an
   address taken from the ACP as the network layer source address, the
   Client's link-layer address as the link-layer source address, and the
   Server's Internet-facing link-layer address as the link-layer
   destination address. The Server will then rewrite the link-layer
   source address with the Server's own enterprise-facing link-layer
   address and rewrite the link-layer destination address with the
   target AERO node's link-layer address, and the packets will enter the
   enterprise network as though they were sourced from a node located
   within the enterprise. In the reverse direction, when a packet
   sourced by a node within the enterprise network uses a destination
   address from the Client's ACP, the packet will be delivered to the
   security gateway which then rewrites the link-layer destination
   address to the Client's link-layer address and rewrites the link-
   layer source address to the Server's Internet-facing link-layer
   address. The Server then delivers the packet across the VPN to the
   AERO Client. In this way, the AERO virtual link is essentially
   extended *through* the security gateway to the point at which the VPN
   link and AERO link are effectively grafted together by the link-layer
   address rewriting performed by the security gateway. All AERO
   messaging services (including route optimization and mobility
   signaling) are therefore extended to the Client.

   In order to support this virtual link grafting, the security gateway
   (acting as an AERO Server) must keep static neighbor cache entries
   for all of its associated Clients located on the public Internet.
   The neighbor cache entry is keyed by the AERO Client's AERO address
   the same as if the Client were located within the enterprise
   internetwork. The neighbor cache is then managed in all ways as
   though the Client were an ordinary AERO Client. This includes the
   AERO IPv6 ND messaging signaling for Route Optimization and Neighbor
   Unreachability Detection.

   Note that the main difference between a security gateway acting as an
   AERO Server and an enterprise-internal AERO Server is that the
   security gateway has at least one enterprise-internal physical
   interface and at least one public Internet physical interface.
   Conversely, the enterprise-internal AERO Server has only enterprise-
   internal physical interfaces. For this reason security gateway
   proxying is needed to ensure that the public Internet link-layer
   addressing space is kept separate from the enterprise-internal link-
   layer addressing space. This is afforded through a natural extension
   of the security association caching already performed for each VPN
   client by the security gateway.

Appendix E. Change Log

   Changes from -77 to -78:

   o Added "V" bit to SLLAO flags field for NS messages. V=1 indicates
      that the NA response must go through the reverse chain of Servers
      and Relays

   o Now using combined DHCPv6/IPvND messages (DHCPv6 as an IPv6ND
      message option)

   o Clarified the use of the "P" bit in the RA flags field

   o Use of SEND to protect the combined DHCPv6/IPv6ND messages

   o Proxy now treats a Client's Servers as the default routers (i.e.,
      instead of using a Relay as the default).

   Changes from -76 to -77:

   o Now using IPv6 ND NS/NA messaging for route optimization (no
      longer using Predirect/Redirect)

   o Now using combined IPv6 ND/DHCPv6 messaging so autoconfiguration
      can be conducted in a single message exchange

   o Introduced the AERO Proxy construct. Critical for applications
      such as ATN/IPS

   Changes from -75 to -76:

   o Bumped version number ahead of expiration deadline

   Changes from -74 to -75:

   o Bumped version number ahead of expiration deadline

Author's Address

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

   Email: fltemplin@acm.org