Network Working Group F. Templin, Ed.
Internet-Draft Boeing Research & Technology
Obsoletes: rfc5320, rfc5558, rfc5720, February 27, 2019
rfc6179, rfc6706 (if
approved)
Intended status: Standards Track
Expires: August 31, 2019

Asymmetric Extended Route Optimization (AERO)
draft-templin-intarea-6706bis-06.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 ND to IP forwarding. Dynamic link selection, mobility management, quality of service (QoS) signaling and route optimization are naturally supported through dynamic neighbor cache updates, while IPv6 Prefix Delegation (PD) is supported by network services such as the Dynamic Host Configuration Protocol for IPv6 (DHCPv6). AERO is a widely-applicable tunneling solution especially well-suited to aviation services, mobile Virtual Private Networks (VPNs) and other applications as described in this document.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 31, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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 ND to IP forwarding. Dynamic link selection, mobility management, quality of service (QoS) signaling and route optimization are naturally supported through dynamic neighbor cache updates, while IPv6 Prefix Delegation (PD) is supported by network services such as the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC8415].

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

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

2. Terminology

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

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

IPv6 Prefix Delegation (PD)
a networking service for delegating IPv6 prefixes to nodes on the link. The nominal PD service is DHCPv6 [RFC8415], however alternate services (e.g., based on ND messaging) are also in scope [I-D.templin-v6ops-pdhost][I-D.templin-6man-dhcpv6-ndopt].

(native) Internetwork
a connected IP network topology over which the AERO link virtual overlay is configured and native peer-to-peer communications are supported. Example Internetworks include the global public Internet, private enterprise networks, aviation networks, etc.

AERO link
a Non-Broadcast, Multiple Access (NBMA) tunnel virtual overlay configured over an underlying Internetwork. 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 Internetwork hops. The AERO mechanisms can also

operate over native link types (e.g., Ethernet, WiFi etc.) when tunneling 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 PDs from one or more AERO Servers. Following PD, the Client assigns a Client AERO address to the AERO interface for use in 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 and a Mobility Anchor Point (MAP) for AERO Clients. The Server assigns an administratively-provisioned AERO address to the AERO interface to support the operation of the ND/PD services. An AERO Server can also act as an AERO Relay.

AERO Relay ("Relay")
an IP router that can relay IP packets between AERO Servers and/or forward IP packets between the AERO link and the native Internetwork. AERO Relays are standard IP routers that do not require any AERO-specific functions.

AERO Proxy ("Proxy")
a node that provides proxying services, 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 external Internetwork in the same manner as for common web proxies, and behaves in a similar fashion as for ND 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
the same as defined for Internetwork.

underlying link
a link that connects an AERO node to the underlying network.

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

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. Packets transmitted over an AERO interface use link-layer addresses as encapsulation header source and destination addresses. Destination link-layer addresses can be either "reachable" or "unreachable" based on dynamically-changing network conditions.

network layer address
the source or destination address of an 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.

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

secured enclave

a private access network (e.g., a corporate enterprise network, radio access network, cellular service provider network, etc.) with secured links and perimeters. Link-layer security services such as IEEE 802.1X and physical-layer security such as campus wired LANs prevent unauthorized access from within the enclave, while border network-layer security services such as firewalls and proxies prevent unauthorized access from the external Internetwork.

Potential Router List (PRL)
a geographically and/or topologically referenced list of IP addresses of Servers for the AERO link.

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

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

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 [RFC8415].

The terminology of DHCPv6 [RFC8415] and IPv6 ND [RFC4861] (including the names of node variables, messages and protocol constants) is used throughout this document. Also, the term "IP" is used to generically refer to either Internet Protocol version, i.e., IPv4 [RFC0791] or IPv6 [RFC8200].

The terms Mobility Anchor Point (MAP) and Distributed Mobility Management (DMM) are used in the same sense as standard Internetworking terminology.

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

```
                              .-(::::::::)
                           .-(::::::::::::)-.
                          (:: 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    |
     +------+-------+     .--------+------.      +-----+--------+
            |           (- Proxyed Clients -)          |
           .-.            `---------------'           .-.
        ,-(  _)-.                                  ,-(  _)-.
     .-(_  IP   )-.   +-------+     +-------+    .-(_  IP   )-.
   (__    EUN      )--|Host H1|     |Host H2|--(__    EUN      )
      `-(______)-'    +-------+     +-------+     `-(______)-'
```

Figure 1: AERO Link Reference Model

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

- o 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 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 other AERO link neighbors.

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

3.2. AERO Node Types

AERO Relays are standard IP routers that provide default forwarding services for AERO Servers. Each Relay also peers with Servers and other Relays in a dynamic routing protocol instance to provide a Distributed Mobility Management (DMM) service for the list of active 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 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 tunnels with neighboring Servers, and maintain an IP forwarding table entry for each AERO Client Prefix (ACP).

AERO Servers provide default forwarding and Mobility Anchor Point (MAP) services for 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 facilitate PD exchanges with Clients, where 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, and track each Client's mobility profiles.

AERO Clients act as requesting routers to receive ACPs through 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 transparent conduit for AERO Clients connected to secured enclaves to associate with AERO Servers. 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. The Proxy forwards the Client's control 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 3.15.

AERO Relays, Servers and Proxies are critical infrastructure elements in fixed (i.e., non-mobile) deployments. Relays, Servers and Proxies must use public link-layer addresses that do not change and can be reached from any correspondent in the underlying Internetwork (i.e., in the same fashion as for popular Internet services). AERO Clients may be mobile, and may not have any public link-layer addresses, e.g., if they are located behind NATs or Proxies.

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

In a reference deployment, each Server is configured as an Autonomous System Border Router (ASBR) for a stub Autonomous System (AS) using an AS Number (ASN) that is unique within the BGP instance, and each Server further uses eBGP to peer with one or more Relays but does not peer with other Servers. 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 only 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. As of 2015, the global public Internet BGP routing system manages more than 500K routes with linear growth and no signs of router resource exhaustion [BGP]. More recent network emulation studies have also shown that a single Relay can accommodate at least 1M dynamically changing BGP routes even on a lightweight virtual machine, i.e., and without requiring high-end dedicated router hardware.

Therefore, assuming each Relay can carry 1M or more routes, this means that at least 1M Clients can be serviced by a single set of Relays. A means of increasing scaling would be to assign a different set of Relays for each set of 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 Internetwork 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.

In an alternate routing arrangement, each set of Relays could advertise an aggregated ASP for the link into the native Internetwork 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.

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

3.4. AERO Addresses

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

For IPv6, Client AERO addresses begin with the prefix fe80::/64 and include in the interface identifier (i.e., the lower 64 bits) a 64-bit prefix taken from one of the Client's IPv6 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, Client AERO addresses are based on an IPv4-mapped IPv6 address 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

Relay, Server and Proxy AERO addresses are allocated from the range fe80::/96, and MUST be managed for uniqueness by the administrative authority for the link. For interfaces that assign static IPv4 addresses, the lower 32 bits of the AERO address includes the IPv4 address, e.g., for the IPv4 address 192.0.2.1 the corresponding AERO address is fe80::192.0.2.1. For other interfaces, the lower 32 bits of the AERO address includes a unique integer value, e.g., fe80::1, fe80::2, fe80::3, etc. The address fe80:: is reserved as the IPv6 link-local Subnet Router Anycast address [RFC4291], and the address fe80::ffff:ffff is reserved as the unspecified AERO address; hence, these values are not available for administrative assignment. (Note that a special link-local-format unspecified address is defined for AERO to satisfy PD services that require a link-local source address.)

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

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

AERO addresses that embed an IPv6 prefix can be statelessly transformed into an IPv6 Subnet Router Anycast address and vice-versa. For example, for the AERO address fe80::2001:db8:2000:3000 the corresponding Subnet Router Anycast address is 2001:db8:2000:3000::. In the same way, for the IPv6 Subnet Router Anycast address 2001:db8:1:2:: the corresponding AERO address is fe80::2001:db8:1:2. In other words, the low-order 64 bits of an AERO address can be used as the high-order 64 bits of a Subnet Router Anycast address, and vice-versa.

AERO links additionally require a reserved IPv6 prefix to support encapsulated forwarding of IPv6 ND messages between Servers on the

link. Although any non-link-local IPv6 prefix could be reserved for this purpose, a Unique Local Address (ULA) prefix [RFC4389] would be desireable since it is not routable outside of the AERO link. For example, if the reserved (ULA) prefix is fd00:db8::/64 the AERO Server Subnet Router Anycast Address is fd00:db8::.

A full discussion of the AERO addressing service is found in [I-D.templin-6man-aeroaddr].

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 ND messages including Router Solicitation (RS), Router Advertisement (RA), Neighbor Solicitation (NS), Neighbor Advertisement (NA) and Redirect for neighbor cache management.

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

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

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

- Prefix Length is set to the ACP prefix length if the ND message source address is a Client AERO address, or 128 if the source is an administratively-provisioned AERO address. For example, if the source address is fe80::2001:db8:1:0 and the ACP is 2001:db8:1:0::/56, Prefix Length is set to the value 56. Conversely, if the source address is fe80::1, Prefix Length is set to 128. If the ND message contains multiple S/TLLAOs, only the Prefix Length value in the first S/TLLAO is consulted and the values in other S/TLLAOs are ignored.

- The 'X' bit is set to '1' in the SLLAO of RS/RA messages by the Proxy when there is a Proxy in the path; otherwise, set to '0'. If the ND message contains multiple SLLAOs, only the 'X' value in

the first SLLAO is consulted and the values in other SLLAOs are ignored.

- The 'N' bit is set to '1' in the SLLAO of RA messages by the Server if there is a NAT in the path; otherwise, set to '0'. If the ND message contains multiple SLLAOs, only the 'N' value in the first SLLAO is consulted and the values in other SLLAOs are ignored.

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

- Interface ID is set to a 16-bit integer value corresponding to an underlying interface of the AERO node. Once the node has assigned an Interface ID to an underlying interface, the assignment must remain unchanged until the node fully detaches from the AERO link. The value '255' is reserved as the AERO Server interface ID, i.e., Servers MUST use Interface ID '255', and Clients MUST number their Interface IDs with values in the range of 0-254.

- UDP Port Number and IP Address are set to the addresses used by the AERO node when it sends encapsulated packets over the specified underlying interface (or to '0' when the addresses are left unspecified). When UDP is not used as part of the encapsulation, UDP Port Number is set to '0'. When the encapsulation IP address family is IPv4, IP Address is formed as an IPv4-mapped IPv6 address as specified in Section 3.4.

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

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

A Client's underlying interfaces are classified as follows:

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

- o NATed interfaces connect to a closed network that is separated from the open Internetwork by a Network Address Translator (NAT). The NAT does not participate in any AERO control message signaling, but the AERO Server can issue control messages on behalf of the Client.

- o VPNed interfaces use security encapsulation over the Internetwork to a Virtual Private Network (VPN) gateway that also acts as an AERO Server. As with NATed links, the AERO Server can issue control messages on behalf of the Client.

- o Proxyed interfaces connect to a closed network that is separated from the open Internetwork by an AERO Proxy. Unlike NATed and VPNed interfaces, the AERO Proxy can also issue control messages on behalf of the Client.

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

If a Client's multiple underlying interfaces are used "one at a time" (i.e., all other interfaces are in standby mode while one interface is active), then ND 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 ND it would appear to have multiple link-layer addresses. In that case, ND messages MAY include multiple S/TLLAOs -- each with an Interface ID that corresponds to a specific underlying interface of the AERO node.

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

When an ND message includes multiple S/TLLAOs, the first S/TLLAO MUST correspond to the AERO node'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 AERO address fe80::ID to the interface. Each fe80::ID address MUST be unique among all AERO nodes on the link. The Relay then engages in a dynamic routing protocol session with one or more Servers and all other Relays on the link (see: Section 3.3), and advertises its assigned ASPs into the native Internetwork. Each Relay subsequently maintains an IP forwarding table entry for each active ACP covered by its ASP(s).

3.6.2. AERO Server Behavior

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

When the Server receives an NS/RS message on the AERO interface it authenticates the message and returns an NA/RA message. (When the Server receives an unsolicited NA message, it likewise authenticates the message and processes it locally.) 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 at the link layer without ever disturbing the network layer.

3.6.3. AERO Proxy Behavior

When a Proxy enables an AERO interface, it assigns an administratively-provisioned address fe80::ID the same as for Relays and Servers. The Proxy further maintains per-Client proxy neighbor cache entries based on control message exchanges. Proxies forward packets between their associated Clients and each Client's associated Servers.

When the Proxy receives an RS message from a Client in the secured enclave, it creates an incomplete proxy neighbor cache entry and sends a proxyed RS 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 completes the proxy neighbor

cache entry based on autoconfiguration information in the RA and sends a proxyed 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 proxy neighbor association.

3.6.4. AERO Client Behavior

When a Client enables an AERO interface, it sends RS messages with ND/PD parameters over an underlying interface to one or more AERO Servers, which return RA messages with corresponding PD parameters. (The RS/RA messages may pass through a Proxy on the path in the case of a Client's Proxyed interface.) See [I-D.templin-6man-dhcpv6-ndopt] for the types of ND/PD parameters that can be included in the RS/RA message exchanges.

After the initial ND/PD message exchange, the Client assigns AERO addresses to the AERO interface based on the delegated prefix(es). The Client can then register additional underlying interfaces with the Server by sending a simple RS message (i.e., one with no PD parameters) over each underlying interface using its base AERO address as the source network layer address. The Server will update its neighbor cache entry for the Client and return a simple 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 ND messages on the AERO interface it updates or creates neighbor cache entries, including link-layer address and QoS preferences.

3.7. AERO Interface Neighbor Cache Maintenance

Each AERO interface maintains a conceptual neighbor cache that includes an entry for each neighbor it communicates with on the AERO link, the same as for any IPv6 interface [RFC4861]. AERO interface neighbor cache entries are said to be one of "permanent", "static", "proxy" 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 their associated Relays and Servers on the link, and AERO Servers maintain permanent neighbor cache entries for their associated 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 ND/PD exchanges as specified in Section 3.14, and remain in place for durations bounded by ND/PD lifetimes. AERO Servers maintain static neighbor cache entries for each of their associated Clients, and AERO Clients maintain static neighbor cache entries for each of their associated Servers.

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

Dynamic neighbor cache entries are created or updated based on receipt of route optimization messages as specified in Section 3.16, and are garbage-collected when keepalive timers expire. AERO route optimization sources maintain dynamic neighbor cache entries for each of their active target Clients with lifetimes based on ND messaging constants.

When a target AERO Server (acting as a Mobility Anchor Point (MAP)) receives a valid NS message used for route optimization, it searches for a static neighbor cache entry for the target Client. The Server then returns an NA message, and adds the link-layer address of the source to a "Report List" associated with the Client's static neighbor cache entry. The Server then sets a "ReportTime" variable for the Report list entry to REPORTTIME seconds. The Server resets ReportTime when it receives a new NS message, and otherwise decrements ReportTime while no NS messages have been received. It is RECOMMENDED that REPORTTIME be set to the default constant value 40 seconds to allow a 10 second window so that the AERO route optimization procedure can converge before ReportTime decrements below REACHABLETIME (see below).

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

The route optimization source also uses the value MAX_UNICAST_SOLICIT to limit the number of NS keepalives sent when a correspondent may have gone unreachable, the value MAX_RTR_SOLICITATIONS to limit the

number of RS messages sent without receiving an RA and the value MAX_NEIGHBOR_ADVERTISEMENT to limit the number of unsolicited NAs that can be sent based on a single event. It is RECOMMENDED that MAX_UNICAST_SOLICIT, MAX_RTR_SOLICITATIONS and MAX_NEIGHBOR_ADVERTISEMENT be set to 3 the same as specified in [RFC4861].

Different values for REPORTTIME, REACHABLETIME, MAX_UNICAST_SOLICIT, MAX_RTR_SOLCITATIONS and MAX_NEIGHBOR_ADVERTISEMENT MAY be administratively set; however, if different values are chosen, all nodes on the link MUST consistently configure the same values. Most importantly, REPORTTIME SHOULD be set to a value that is sufficiently longer than REACHABLETIME to allow the AERO route optimization procedure to converge.

When there may be a NAT or Proxy between the Client and the Server, or if the path from the Client to the Server should be tested for reachability, either the Client or the Proxy can send periodic RS messages to the Server without PD parameters to receive RA replies. The RS/RA messaging will keep NAT/Proxy state alive and test Server reachability without disturbing the PD service.

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 tunneled 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 QoS preferences, and also to select a destination link-layer address based on the neighbor's underlying interface with the highest preference. AERO implementations SHOULD allow for QoS preference values to be modified at runtime through network management.

If multiple outgoing interfaces and/or neighbor interfaces have a preference of "high", the AERO node sends one copy of the packet via

each of the (outgoing / neighbor) interface pairs; otherwise, the node sends a single copy of the packet via the interface with the highest preference. AERO nodes keep track of which underlying interfaces are currently "reachable" or "unreachable", and only use "reachable" interfaces for forwarding purposes.

The following sections discuss the AERO interface forwarding algorithms for Clients, 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 AERO 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 "reachable" 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 there may 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 and MAY return a network-layer ICMP Destination Unreachable message subject to rate limiting (see: Section 3.13).

3.8.2. Proxy Forwarding Algorithm

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

When the Proxy receives a packet from an AERO interface neighbor, it searches for a proxy neighbor cache entry for a Client within the secured enclave that matches the destination. If there is a match, the Proxy forwards the packet to the Client. Otherwise, the Proxy returns the packet to the neighbor, i.e., by reversing the source and destination link-layer addresses and re-admitting the packet into the AERO link.

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 static neighbor cache entry for a Client 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 permanent 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 according to the network-layer destination address as follows:

- 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.

- else, if the destination matches a static neighbor cache entry for a local Client the Server first determines whether the neighbor is the same as the one it received the packet from. If so, the Server drops 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.

- else, if the destination matches a dynamic neighbor cache entry for a target Client, the Server forwards the packet according to the interface ID settings in the dynamic neighbor cache entry.

- 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

Relays forward packets the same as any IP router. When the Relay receives an encapsulated packet from a Server via the AERO link, it removes the encapsulation header and searches for a forwarding table entry that matches the destination address in the next IP header. When the Relay receives an unencapsulated packet from a node outside the AERO link, it performs the same forwarding table lookup. The Relay then processes the packet as follows:

- if the destination does not match an ASP, or if the destination matches one of the Relay's own addresses, the Relay submits the packet for either IP forwarding or local delivery.

- else, if the destination matches an ACP entry in the IP forwarding table 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 encapsulates and forwards the packet using the neighbor's link-layer address as the destination for encapsulation.

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

As for any IP router, the Relay decrements the TTL/Hop Count when it forwards the packet.

3.8.5. Processing Return Packets

When an AERO Server receives a return packet from an AERO Proxy (see Section 3.8.2), it proceeds according to the AERO link trust basis. Namely, the return packets have the same trust profile as for link-layer Destination Unreachable messages. If the Server has sufficient trust basis to accept link-layer Destination Unreachable messages, it can then process the return packet by searching for a dynamic neighbor cache entry that matches the destination. If there is a match, the Server marks the corresponding link-layer address as "unreachable", selects the next-highest priority "reachable" link-layer address in the entry as the link-layer address for encapsulation then (re)admits the packet into the AERO link. If there are no "reachable" link-layer addresses, the Server instead sets ReachableTime in the dynamic neighbor cache entry to 0. Otherwise, the Server SHOULD drop the packet and treat it as an indication that a path may be failing, and MAY use Neighbor Unreachability Detection (NUD) (see: Section 3.13) to test the path for reachability.

When an AERO Relay receives a return packet from an AERO Server, it searches its forwarding table for an entry that matches the inner destination address. If there is a forwarding table entry that lists a different Server as the next hop, the Relay forwards the packet to the different Server; otherwise, the Relay drops the packet.

See Section 3.18 for further discussion on the nature of return packets.

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-intarea-gue][I-D.ietf-intarea-gue-extensions], or through an alternate encapsulation format (e.g., see: Appendix A, [RFC2784], [RFC8086], [RFC4301], etc.). 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.

Clients normally use the IP address of the underlying interface as the encapsulation source address. If the underlying interface does not have an IP address, however, the Client uses an IP address taken from an ACP as the encapsulation source address (assuming the node has some way of injecting the ACP into the underlying network routing system). For IPv6 addresses, the Client normally uses the ACP Subnet Router Anycast address [RFC4291].

When GUE encapsulation is not available, encapsulation between Servers and Relays can use standard mechanisms such as Generic Routing Encapsulation (GRE) [RFC2784], GRE-in-UDP [RFC8086] and IPSec [RFC4301] so that Relays can be standard IP routers with no AERO-specific mechanisms.

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 ND messages from Clients (either directly or via a Proxy), and create or update a static neighbor cache entry for the Client based on the specific message type.

- o AERO Clients 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 Proxies accept encapsulated packets if there is a proxy neighbor cache entry that matches the packet's network-layer address.

Each packet should 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 some environments, however, it may be sufficient to require signatures only for ND control plane messages (see: Section 10) 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 [RFC8200]. Although IPv4 specifies a smaller minimum link MTU of 68 bytes [RFC0791], AERO interfaces also observe the IPv6 minimum for IPv4 even if encapsulated packets may incur network fragmentation.

IPv6 specifies a minimum Maximum Reassembly Unit (MRU) of 1500 bytes [RFC8200], while the minimum MRU for IPv4 is only 576 bytes [RFC1122] (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.ietf-intarea-gue-extensions]) 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 in the underlying network 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 process the message as an indication that a path may be failing, and MAY initiate NUD over that path. If it receives Destination Unreachable messages on many or all paths, the node SHOULD set ReachableTime for the corresponding dynamic neighbor

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

- 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 mark the path as unusable and use another path. If it receives Destination Unreachable messages on many or all paths, the Client SHOULD associate with a new Server and release its association with the old Server as specified in Section 3.18.7.

- 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 mark the underlying path as unusable and use another underlying path. If it receives Destination Unreachable messages on multiple paths, the Server should take no further actions unless it receives an explicit ND/PD release message or if the PD lifetime expires. In that case, the Server MUST release the Client's delegated ACP, withdraw the ACP from the AERO routing system and delete the neighbor cache entry.

- 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, the dynamic routing protocol should soon reconverge and correct the temporary outage.

When an AERO Relay receives a packet for which the network-layer destination address is covered by an 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 writes the network-layer source address of the original packet as the destination address and uses one of its non link-local addresses as the source address of the message.

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

3.14. AERO Router Discovery, Prefix Delegation and Autoconfiguration

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

3.14.1. AERO ND/PD Service Model

Each AERO Server configures a PD service to facilitate Client requests. 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 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 releasing PDs received from existing Servers. This provides the Client with a natural fault-tolerance and/or load balancing profile.

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

AERO Clients and Servers include PD parameters in the RS/RA messages they exchange (see: [I-D.templin-6man-dhcpv6-ndopt]). The unified ND/PD messages are exchanged between Client and Server according to the prefix management schedule required by the PD service. If the Client knows its ACP in advance, it can include its AERO address as the source address of an RS message and with an SLLAO with a valid Prefix Length for the ACP. If the Server (and Proxy) accept the Client's ACP assertion, they inject the prefix into the routing system and establish the necessary neighbor cache state. If the Client does not know its ACP in advance, or if it wishes to engage in a formal PD exchange, it can use a service such as DHCPv6.

On Some AERO links, PD arrangements may be through some out-of-band service such as network management, static configuration, etc. In those cases, AERO nodes can use simple RS/RA message exchanges with no explicit PD options. Instead, the RS/RA messages use AERO addresses as a means of representing the delegated prefixes, e.g., if a message includes a source address of "fe80::2001:db8:1:2" then the recipient can infer that the sender holds the prefix delegation

"2001:db8:1:2::/N" (where 'N' is the Prefix Length included in the first S/TLLAO in the message).

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 in the Potential Router List (PRL) via static configuration (e.g., from a flat-file map of Server addresses and locations), or through an automated means such as Domain Name System (DNS) name resolution [RFC1035]. In the absence of other information, the Client resolves the DNS Fully-Qualified Domain Name (FQDN) "linkupnetworks.[domainname]" where "linkupnetworks" is a constant text string and "[domainname]" is a DNS suffix for the Client's underlying interface (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 an ND/PD message exchange. The Client sends an RS message with PD parameters and with 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. If the Client already knows its own AERO address, it uses the AERO address as the IPv6 source address; otherwise, it uses the unspecified AERO address as the source address. If the Client's underlying interface connects to a subnetwork that supports ACP injection, the Client can use the ACP's Subnet Router Anycast address as the link-layer source address.

The Client next includes one or more SLLAOs in the RS message 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 message. The Client MAY include additional SLLAOs specific to other underlying interfaces, but if so it sets their UDP Port Number and IP Address fields to 0. The Client can instead register additional link-layer addresses with the Server by sending additional RS messages including SLLAOs via other underlying interfaces after the initial RS/RA exchange.

The Client then sends the RS message to the AERO Server and waits for an RA message reply (see Section 3.14.3) while retrying MAX_RTR_SOLICITATIONS times until an RA is received. If the Client receives no RAs, or if it receives an RA with Router Lifetime set to 0 and/or with no ACP PD parameters, the Client SHOULD discontinue autoconfiguration attempts through this Server and try another

Server. Otherwise, the Client processes the ACPs found in the RA message.

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

The Client next examines the X and N bits in the first SLLAO of the RA message. If the X bit value is '1' the Client infers that there is a Proxy on the path via the interface over which it sent the RS message, and if the N bit value is '1' the Client infers that there is a NAT on the path. If N is '1', the Client sets UDP Port Number and IP Address to 0 in the first S/TLLAO of any subsequent ND messages it sends to the Server over that link.

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 receive 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] to support the Client's downstream attached "Internet of Things (IoT)". The Client subsequently maintains its ACP delegations through each of its Servers by sending RS messages with PD parameters to receive corresponding RA 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 changes address or becomes unavailable, if QoS preferences change, etc. To do so, the Client prepares an RS message to send over any available underlying interface. The RS MUST include a SLLAO specific to the selected available underlying interface as the first SLLAO and MAY include any additional SLLAOs specific to other underlying interfaces. The Client includes fresh 'P(i)' values in each SLLAO to update the Server's neighbor cache entry. If the Client wishes to update 'P(i)' values without updating the link-layer address, it sets the UDP Port Number and IP Address fields to 0. If the Client wishes to disable the interface, it sets all 'P(i)' values to '0' ("disabled"). When the Client receives the Server's RS response, it has assurance that the Server has been updated with the new information.

If the Client wishes to discontinue use of a Server it issues an RS message with PD parameters that will cause the Server to release the Client. When the Server processes the message, it releases the ACP, deletes its neighbor cache entry for the Client, withdraws the IP route from the routing system and returns an RA reply containing any necessary PD parameters.

3.14.3. AERO Server Behavior

AERO Servers act as IPv6 routers and support a PD service for Clients. 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. The list of Server addresses should be geographically and/or topologically referenced, and forms the Potential Router List (PRL) for the AERO link.

When an AERO Server receives a prospective Client's RS message with PD parameters on its AERO interface, and the Server is too busy, it SHOULD return an immediate RA reply with no ACPs and with Router Lifetime set to 0. Otherwise, the Server authenticates the RS message and processes the PD parameters. 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 an RA message that includes the delegated ACPs, any other PD parameters and an SLLAO with the Server's link-layer address and with Interface ID set to 0. The Server then returns the RA message using its link-local address as the network-layer source address, the network-layer source address of the RS message 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 N flag to 1 if 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 on the path. 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 sends the RA message 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 in the RS 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 in the path. The Server also records the value of the X bit to indicate whether there is a Proxy on the path.

After the initial RS/RA exchange, the AERO Server maintains the neighbor cache entry for the Client until the PD lifetimes expire. If the Client (or Proxy) issues additional RS messages with PD renewal parameters, the Server extends the PD lifetimes. If the Client (or Proxy) issues an RS with PD release parameters, or if the Client (or Proxy) does not issue a renewal before the lifetime expires, the Server deletes the neighbor cache entry for the Client and withdraws the IP routes from the AERO routing system. The Server processes these and any other Client PD messages, and returns an RA reply. The Server may also issue an unsolicited RA message with PD reconfigure parameters to inform the Client that it needs to renegotiate its PDs.

3.14.3.1. Lightweight DHCPv6 Relay Agent (LDRA)

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

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

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

When the DHCPv6 server prepares a Reply message, it wraps the message in a 'Relay-Reply' message and echoes the Interface-Id option. The DHCPv6 server then delivers the Relay-Reply message to the LDRA, which discards the Relay-Reply wrapper and IPv6/UDP headers, then uses the DHCPv6 message to fabricate 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. The AERO Proxy

In some environments, Clients may be located in secured subnetwork enclaves that do not allow direct communications from the Client to a Server in the outside Internetwork. In that case, the secured enclave can employ an AERO Proxy.

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

o when the Proxy receives an RS message from a Client within the secured enclave, it first authenticates the message then creates a proxy neighbor cache entry for the Client in the INCOMPLETE State and caches the Client and Server link-layer addresses along with any identifying information including Transaction IDs, Client Identifiers, Nonce values, etc. The Proxy then re-encapsulates the RS message using its own external address as the source link-layer address, sets the X flag in the first SLLAO to '1', and forwards the message to the Server.

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

o when the Proxy receives the RA message, it matches the message with the RS that created 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 re-encapsulates the RA message using its own internal address as the source link-layer address, sets the X flag in the first SLLAO to '1', and forwards the message to the Client.

After the initial RS/RA exchange, the Proxy forwards data packets between the Client and Server with the Server acting as the Client's

default router. The Proxy can send RS messages to the Client's Server(s) to update Server neighbor cache entries on behalf of the Client, e.g., to refresh neighbor cache entry lifetimes and/or to convey QoS updates. The Proxy also forwards any Client control and data messages to the Client's primary Server.

In some subnetworks that employ a Proxy, the Client's ACP can be injected into the underlying network routing system. In that case, the Client can send data messages without encapsulation so that the native underlying network routing system transports the unencapsulated packets to the Proxy. This can be very beneficial, e.g., if the Client connects to the network via low-end data links such as some aviation wireless links. In that case, however, the Client's control messages are still sent encapsulated so as to supply the Proxy with the address of the Server and to transport IPv6 ND messages without decrementing the hop-count. In summary, the interface becomes one where control messages are encapsulated while data messages are either unencapsulated or encapsulated according to the specific use case. This encapsulation avoidance represents a form of "header compression", meaning that the MTU should be sized based on the size of full encapsulated messages even if most messages are sent unencapsulated.

3.16. AERO Route Optimization

While data packets are flowing from a source Client to a target Client that are both holders of ACPs belonging to the same AERO link, route optimization SHOULD be used to avoid sending traffic through sub-optimal routes that consume expensive resources. Route optimization is conducted on a per-interface basis based on the source Client's available underlying interfaces, and may need to involve Proxies and Servers in the process.

Route optimization is initiated by the first eligible AERO node closest to the source Client (i.e., the route optimization source) as follows:

o For VPNed, NATed and Direct underlying interfaces, the Server is the route optimization source.

o For Proxyed underlying interfaces, the Proxy is the route optimization source.

o For native underlying interfaces, the Client itself is the route optimization source.

While the source Client sends data packets toward a target Client, the route optimization source also sends an NS message to receive a

solicited NA message from a target Server acting as a Mobility Anchor Point (MAP) for the target Client. The route optimization process parallels IPv6 ND Address Resolution.

The NS includes the AERO address of the route optimization source as the network-layer source address, the AERO address corresponding to the data packet's destination address as the network-layer destination address, and the route optimization source address as the link-layer source address. For Clients and Proxies as the route optimization source, the address of the Client's Server is used as the link-layer destination address. For Servers as the route optimization source, the address of a Relay is used as the link-layer destination address. The NS message also includes a single SLLAO with the route optimization source address in the UDP Port Number and IP address fields. Finally, the NS message includes a Timestamp and Nonce option that can be used to match against the corresponding solicited NA.

When the source Server receives or originates the NS message, it inserts an additional mid-layer IP encapsulation header between the NS message link-layer and network-layer headers. This mid-layer IP header uses the AERO Server Subnet Router Anycast address as the source address and the Subnet Router Anycast address corresponding to the target AERO address as the destination address. The source Server then changes the link-layer source address to its own address and the link-layer destination address to the address of a Relay. The Server finally forwards the message to the Relay without decrementing the network-layer TTL/Hop Limit field.

When the Relay receives the double-encapsulated NS message from the source Server, it discards the link-layer header(s) and determines that the target Server is the next hop toward the target Client by consulting its standard IP forwarding table for the Client Subnet Router Anycast destination address. The Relay then encapsulates and forwards the message to the target Server the same as for any IP router.

When the target Server receives the double-encapsulated NS message from the Relay, it removes the link-layer and mid-layer headers and examines the network-layer destination address to determine whether the target Client is one of its static neighbors. If the target Client is not a static neighbor, the target Server discards the NS and prepares an NA message with no TLLAOs to send back to the route optimization source. The target Server then encapsulates the NA message and sets the link-layer source address to its own address and sets the link-layer destination address to the address found in the NS SLLAO. The target Server finally includes the Nonce value

received in the NS plus the current Timestamp, then sends the NA message to the route optimization source.

Otherwise, the target Server adds the link-layer address found in the NS SLLAO to the "Report" list for the target Client's neighbor cache entry with timer set to ReportTime seconds, but it does not create or update a neighbor cache entry. The target Server then prepares an NA message to send back to the route optimization source. The NA message includes a first TLLAO with the target Server's address in the IP address and UDP Port Number, with Interface ID set to '255', with all P(i) values set to "low" and with "Prefix Length" set to the prefix length of the target Client's ACP. The NA message then includes additional TLLAOs for all of the target Client's underlying interfaces. For NATed, VPNed and Direct interfaces, the TLLAO addresses are the address of the Server. For Proxyed interfaces, the TLLAO addresses are the addresses of the Proxies, and for native interfaces the TLLAO addresses are the addresses of the Client. The target Server then encapsulates the NA message and sets the link-layer source address to its own address and sets the link-layer destination address to the address found in the NS SLLAO. The target Server finally includes the Nonce value received in the NS plus the current Timestamp, then sends the NA message to the route optimization source.

When the route optimization source receives the NA message, it verifies the Nonce and Timestamp. If there are no TLLAOs, the route optimization source discards the message; otherwise, it creates a dynamic neighbor cache entry for the target Client and caches all address, Interface ID, P(i) and Prefix Length information found in the NA TLLAOs. The route optimization source also sets the neighbor cache entry state to REACHABLE and sets the timeout value to ReachableTime. Future data packets that flow through the route optimization source will then go directly to the target Client instead of traveling through a dogleg route involving unnecessary Servers and/or Relays. The route optimization further is shared by all sources that send packets to the target Client, i.e., and not just the original source Client.

While new data packets destined to the target are flowing through the route optimization source, it sends additional NS messages to the target Server before ReachableTime expires to receive a fresh NA message. The route optimization source then updates the dynamic neighbor cache entry to refresh ReachableTime, while the target Server updates the target Client's static neighbor cache entry to refresh ReportTime. While no data packets are flowing, the route optimization source instead allows the dynamic neighbor cache entry to expire. Following expiration, future data packets flowing through the route optimization source will again trigger a new route

optimization exchange while initial data packets travel over a suboptimal route via Servers and/or Relays.

In this arrangement, the route optimization source holds a dynamic neighbor cache entry for the target, but the target does not hold a dynamic neighbor cache entry for the route optimization source. The route optimization neighbor relationship is therefore asymmetric and unidirectional. If the target Client also has packets to send back to the source Client, then a separate route optimization procedure is required in the reverse direction. But, there is no requirement that the forward and reverse paths be symmetric.

3.17. Neighbor Unreachability Detection (NUD)

AERO nodes perform Neighbor Unreachability Detection (NUD) by sending NS messages to elicit solicited NA messages from the target node the same as described in [RFC4861]. NUD is performed either reactively in response to persistent link-layer errors (see Section 3.13) or proactively to confirm reachability in the forward direction.

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 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.

Note that AERO nodes may have multiple underlying interface paths toward the target neighbor. In that case, NUD SHOULD be performed over each underlying interface individually and the node should only consider the neighbor unreachable if NUD fails over multiple underlying interface paths.

Underlying interface paths that pass NUD tests are marked as "reachable", while those that do not are marked as "unreachable". These markings inform the AERO interface forwarding algorithm specified in Section 3.8.

3.18. Mobility Management and Quality of Service (QoS)

AERO is an example of a Distributed Mobility Management (DMM) service. Each Server is responsible for only a subset of the Clients on the AERO link, as opposed to a Centralized Mobility Management (CMM) service where there is a single network mobility service for

all Clients. Clients coordinate with their associated Servers via RS/RA exchanges to maintain the DMM profile, and the AERO routing system tracks all current Client/Server peering relationships.

Servers provide a Mobility Anchor Point (MAP) for their dependent Clients. Clients are responsible for maintaining neighbor relationships with their Servers through periodic RS/RA exchanges, which also serves to confirm Client reachability. When a Client's underlying interface address and/or QoS information changes, the Client is responsible for updating the Server with this new information. (Note that for Proxyed interfaces, however, the Proxy can perform the RS/RA exchanges on the Client's behalf.)

Mobility management considerations are specified in the following sections.

3.18.1. Mobility Update Messaging

AERO Servers accommodate mobility and/or QoS change events by sending unsolicited NA messages to all route optimization sources for the Client. When a Server sends an unsolicited NA message, it sets the IPv6 source address to the Client's AERO address, sets the IPv6 destination address to all-nodes multicast, sets the link-layer source address to its own address and sets the link-layer destination address to either a multicast address or the unicast link-layer address of a route optimization source in the Client's Report list. If there are multiple route optimization sources, the node sends identical unicast copies of the unsolicited NA to each source.

As for the hot-swap of interface cards discusssed in Section 7.2.6 of [RFC4861], the transmission and reception of unsolicited NA messages is unreliable but provided as a useful optimization. The Server can send up to MAX_NEIGHBOR_ADVERTISEMENT unsolicited NAs to each route optimization source, but in the normal case sends only one.

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

If unsolicited NA messages are lost, the route optimization source may be left with stale address and/or QoS information for the Client for up to ReachableTime seconds. During this time, the route optimization source can continue sending packets to the target Client according to its current neighbor cache information but may receive

persistent Destination Unreachable messages and/or unsolicited NA messages with no TLLAOs as discussed in Section 3.18.2. In that case, the route optimization source SHOULD re-initiate route optimization immediately instead of waiting for ReachableTime to expire.

3.18.2. 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 to a Relay which delivers them to the target Client's current location. If the source is not one of its static neighbor Clients, the Server also returns an unsolicited NA message to the source with no TLLAOs - the sender will then realize that it needs to delete its neighbor cache entry that associated the target with this Server and re-initiate route optimization.

When a Proxy receives packets with destination addresses that do not match of its proxy neighbor cache Clients, if forwards the packets to a Server which delivers them to the target Client's current location. If the source is not one of its proxy neighbor Clients, the Proxy also returns an unsolicited NA message to the source with no TLLAOs the same as described for Servers above.

When a Client receives packets with destination addresses that do not match one of its ACPs, it drops the packets and returns an unsolicited NA message to the source with no TLLAOs.

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

When a Client needs to change its link-layer addresses and/or QoS preferences (e.g., due to a mobility event), either the Client or Proxy sends RS messages to its Servers using the new link-layer address as the source address and with TLLAOs that include the new Client UDP Port Number, IP Address and P(i) values. If the RS messages are sent solely for the purpose of updating QoS preferences without updating the link-layer address, the UDP Port Number and IP Address are set to 0.

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

3.18.4. 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 RS messages to

its Servers using the new link-layer address as the source address and with TLLAOs that include the new Client link-layer information.

3.18.5. 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 RS messages to its Servers with SLLAOs with all P(i) values set to 0.

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

3.18.6. Implicit Mobility Management

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

If the Client's 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.

3.18.7. Moving to a New Server

When a Client associates with a new Server, it performs the Client procedures specified in Section 3.14.2. The Client then sends RS messages with PD release parameters to the old Server to release itself from that Server's domain. If the Client does not receive an RA reply after MAX_RTR_SOLICITATIONS attempts, the old Server may have failed and the Client should discontinue its release attempts. When the old Server processes the PD release, it sends unsolicited NA messages with no TLLAOs to all route optimization sources in the Client's Report list.

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 no 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, movement to a new geographic region, etc.

3.19. 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. Direct Underlying Interfaces

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

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 message to the Server the same as for ordinary AERO links. The Server responds with an RA message

that includes one or more /128 prefixes (i.e., singleton addresses) that include the /64 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 3.16.

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.

6. AERO Adaptations for SEcure Neighbor Discovery (SEND)

SEcure Neighbor Discovery (SEND) [RFC3971] and Cryptographically Generated Addresses (CGAs) [RFC3972] were designed to secure IPv6 ND messaging in environments where symmetric network and/or transport-layer security services are impractical (see: Section 10). AERO nodes that use SEND/CGA employ the following adaptations.

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

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

7. AERO Critical Infrastructure Considerations

AERO Relays are low-end to midrange Commercial off-the Shelf (COTS) standard IP routers with no AERO code. Relays must be provisioned, supported and managed by the AERO Link Service Provider. Cost for purchasing, configuring and managing Relays is nominal even for very large AERO links.

AERO Servers can be standard dedicated server platforms, but most often will be deployed as virtual machines in the cloud. The only requirements for Servers are that they can run the AERO user-level code and have at least one network interface with a public IP address. As with Relays, Servers must be provisioned, supported and managed by the AERO Link Service Provider. Cost for purchasing, configuring and managing Servers is nominal especially for virtual Servers hosted in the cloud.

AERO Proxies are most often standard dedicated server platforms with one network interface connected to the secured enclave and a second interface connected to the public Internetwork. As with Servers, the only requirements are that they can run the AERO user-level code and have at least one interface with a public IP address. Proxies must be provisioned, supported and managed by the administrative authority for the secured enclave. Cost for purchasing, configuring and managing Proxies is nominal, and borne by the secured enclave administrative authority.

8. 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-2.0.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-4.0.0.tgz.

A survey of public domain and commercial SEND implementations is available at https://www.ietf.org/mail-archive/web/its/current/msg02758.html.

9. 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.

10. Security Considerations

AERO link security considerations include considerations for both the data plane and the control plane.

Data plane security considerations are the same as for ordinary Internet communications. Application endpoints in AERO Clients and their EUNs SHOULD use application-layer security services such as TLS/SSL [RFC5246], DTLS [RFC6347] or SSH [RFC4251] to assure the same level of protection as for critical secured Internet services such as online banking. AERO Clients that require VPN access to enterprise networks SHOULD use symmetric network and/or transport layer security services such as TLS/SSL, DTLS, IPsec [RFC4301], etc.

Control plane security considerations are the same as for standard IPv6 Neighbor Discovery [RFC4861], except that the PRL also improves security by providing AERO Clients with a list of trusted Servers. As fixed infrastructure elements, AERO Proxies and Servers SHOULD pre-configure security associations for one another (e.g., using pre-placed keys) and use symmetric network and/or transport layer security services such as IPsec, TLS/SSL or DTLS to secure ND messages. AERO Clients that connect to secured enclaves need not apply security to their ND messages, since the messages will be intercepted by an enclave perimeter Proxy. AERO Clients located outside of secured enclaves SHOULD use symmetric network and/or transport layer security to secure their ND exchanges with Servers, but when there are many prospective neighbors with dynamically changing connectivity an asymmetric security service such as SEND may be needed (see: Section 6).

AERO Servers and Relays present targets for traffic amplification Denial of Service (DoS) attacks. This concern is no different than for widely-deployed VPN security gateways in the Internet, where attackers could send spoofed packets to the gateways at high data rates. This can be mitigated by connecting 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 and Proxies can institute rate limits that

protect Clients from receiving packet floods that could DoS low data rate links.

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.)

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

The PRL MUST be well-managed and secured from unauthorized tampering, even though the list includes only public information.

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

11. Acknowledgements

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

This work has further been encouraged and supported by Boeing colleagues including Kyle Bae, M. Wayne Benson, Dave Bernhardt, Cam Brodie, Balaguruna Chidambaram, Irene Chin, Bruce Cornish, Claudiu Danilov, Wen Fang, Anthony Gregory, Jeff Holland, Seth Jahne, Ed King, Gene MacLean III, Rob Muszkiewicz, Sean O'Sullivan, Greg Saccone, 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. Kyle Bae, Wayne Benson and Eric Yeh are especially acknowledged for implementing the AERO functions as extensions to the public domain OpenVPN distribution.

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

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

- o The Internet Routing Overlay Network (IRON) [RFC6179][I-D.templin-ironbis]
- o Virtual Enterprise Traversal (VET) [RFC5558][I-D.templin-intarea-vet]
- o The Subnetwork Encapsulation and Adaptation Layer (SEAL) [RFC5320][I-D.templin-intarea-seal]
- o AERO, First Edition [RFC6706]

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

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

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

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

This work is aligned with the Boeing autonomy program.

12. References

12.1. 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>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

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

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

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

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

[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>.

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

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

12.2. Informative References

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

[I-D.ietf-dmm-distributed-mobility-anchoring] Chan, A., Wei, X., Lee, J., Jeon, S., and C. Bernardos, "Distributed Mobility Anchoring", draft-ietf-dmm-distributed-mobility-anchoring-12 (work in progress), January 2019.

[I-D.ietf-intarea-gue] Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-intarea-gue-06 (work in progress), August 2018.

[I-D.ietf-intarea-gue-extensions] Herbert, T., Yong, L., and F. Templin, "Extensions for Generic UDP Encapsulation", draft-ietf-intarea-gue-extensions-05 (work in progress), August 2018.

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

[I-D.ietf-rtgwg-atn-bgp] Templin, F., Saccone, G., Dawra, G., Lindem, A., and V. Moreno, "A Simple BGP-based Mobile Routing System for the Aeronautical Telecommunications Network", draft-ietf-rtgwg-atn-bgp-01 (work in progress), January 2019.

[I-D.templin-6man-aeroaddr] Templin, F., "The AERO Address", draft-templin-6man-aeroaddr-04 (work in progress), December 2018.

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

[I-D.templin-6man-rio-redirect]
Templin, F. and j. woodyatt, "Route Information Options in IPv6 Neighbor Discovery", draft-templin-6man-rio-redirect-07 (work in progress), December 2018.

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

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

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

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

[I-D.templin-v6ops-pdhost]
Templin, F., "IPv6 Prefix Delegation and Multi-Addressing Models", draft-templin-v6ops-pdhost-23 (work in progress), December 2018.

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

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

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

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

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

[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>.

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

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

[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>.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[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>.

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

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

[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>.

[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>.

[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>.

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

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

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

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

[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>.

[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>.

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

[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 4 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 4: Minimal Encapsulation Format using IP-in-IP

Figure 5 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 5: Minimal Encapsulation Using GRE

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

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

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

AERO is especially well suited for use with VPN system encapsulations such as OpenVPN [OVPN].

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

The AERO S/TLLAO format specified in Section 3.5 includes a Length value of 5 (i.e., 5 units of 8 octets). However, special-purpose links may extend the basic format to include additional fields and a Length value larger than 5.

For example, adaptation of AERO to the Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS) includes link selection preferences based on transport port numbers in addition to the existing DSCP-based preferences. ATN/IPS nodes maintain a map of transport port numbers to 64 possible preference fields, e.g., TCP port 22 maps to preference field 8, TCP port 443 maps to preference field 20, UDP port 8060 maps to preference field 34, etc. The extended S/TLLAO format for ATN/IPS is shown in Figure 6, where the Length value is 7 and the 'Q(i)' fields provide link preferences for the corresponding transport port number.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |   Length = 7  | Prefix Length |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Interface ID         |        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|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Q00|Q01|Q02|Q03|Q04|Q05|Q06|Q07|Q08|Q09|Q10|Q11|Q12|Q13|Q14|Q15|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Q16|Q17|Q18|Q19|Q20|Q21|Q22|Q23|Q24|Q25|Q26|Q27|Q28|Q29|Q30|Q31|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Q32|Q33|Q34|Q35|Q36|Q37|Q38|Q39|Q40|Q41|Q42|Q43|Q44|Q45|Q46|Q47|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Q48|Q49|Q50|Q51|Q52|Q53|Q54|Q55|Q56|Q57|Q58|Q59|Q60|Q61|Q62|Q63|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: ATN/IPS Extended S/TLLAO Format

Appendix C. Change Log

<< RFC Editor - remove prior to publication >>

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

- Major re-work and simplification of Route Optimization function
- Added Distributed Mobility Management (DMM) and Mobility Anchor Point (MAP) terminology
- New section on "AERO Critical Infrastructure Element Considerations" demonstrating low overall cost for the service
- minor text revisions and deletions
- removed extraneous appendices

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

- New Appendix E on S/TLLAO Extensions for special-purpose links. Discussed ATN/IPS as example.
- New sentence in introduction to declare appendices as non-normative.

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

- Added definitions for Potential Router List (PRL) and secure enclave
- Included text on mapping transport layer port numbers to network layer DSCP values
- Added reference to DTLS and DMM Distributed Mobility Anchoring working group document
- Reworked Security Considerations
- Updated references.

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

- Added new section on SEND.

- o Clarifications on "AERO Address" section.
- o Updated references and added new reference for RFC8086.
- o Security considerations updates.
- o General text clarifications and cleanup.

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

- o Note on encapsulation avoidance in Section 4.

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

- o Remove DHCPv6 Server Release procedures that leveraged the old way Relays used to "route" between Server link-local addresses
- o Remove all text relating to Relays needing to do any AERO-specific operations
- o Proxy sends RS and receives RA from Server using SEND. Use CGAs as source addresses, and destination address of RA reply is to the AERO address corresponding to the Client's ACP.
- o Proxy uses SEND to protect RS and authenticate RA (Client does not use SEND, but rather relies on subnetwork security. When the Proxy receives an RS from the Client, it creates a new RS using its own addresses as the source and uses SEND with CGAs to send a new RS to the Server.
- o Emphasize distributed mobility management
- o AERO address-based RS injection of ACP into underlying routing system.

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

- o Document use of NUD (NS/NA) for reliable link-layer address updates as an alternative to unreliable unsolicited NA. Consistent with Section 7.2.6 of RFC4861.
- o Server adds additional layer of encapsulation between outer and inner headers of NS/NA messages for transmission through Relays that act as vanilla IPv6 routers. The messages include the AERO Server Subnet Router Anycast address as the source and the Subnet

Router Anycast address corresponding to the Client's ACP as the destination.

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

Author's Address

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

Email: fltemplin@acm.org