Network Working Group
Internet-Draft
Obsoletes: rfc5320, rfc5558, rfc5720, rfc6179, rfc6706 (if approved)
Intended status: Standards Track
Expires: October 1, 2020

F. Templin, Ed.
Boeing Research & Technology

March 30, 2020

Asymmetric Extended Route Optimization (AERO)
draft-templin-intarea-6706bis-35

Abstract

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

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 October 1, 2020.

Table of Contents

1. Introduction

Asymmetric Extended Route Optimization (AERO) fulfills the requirements of Distributed Mobility Management (DMM) [RFC7333] and route optimization [RFC5522] for aeronautical networking and other network mobility use cases. AERO is based on a Non-Broadcast, Multiple Access (NBMA) virtual link model known as the AERO link. The AERO link is a virtual overlay configured over one or more underlying Internetworks, and nodes on the link can exchange IP packets via tunneling. Multilink operation allows for increased reliability, bandwidth optimization and traffic path diversity.

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

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

AERO Relays are interconnected in a secured private BGP overlay routing instance known as the "SPAN". The SPAN provides a hybrid routing/bridging service to join the underlying Internetworks of multiple disjoint administrative domains into a single unified AERO link. Each AERO link instance is characterized by the set of Mobility Service Prefixes (MSPs) common to all mobile nodes. The link extends to the point where a Gateway/Server is on the optimal route from any correspondent node on the link, and provides a gateway between the underlying Internetwork and the SPAN. To the underlying Internetwork, the Gateway/Server is the source of a route to its MSP, and hence uplink traffic to the mobile node is naturally routed to the nearest Gateway/Server.

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

AERO was designed for aeronautical networking for both manned and unmanned aircraft, where the aircraft is treated as a mobile node that can connect an Internet of Things (IoT). AERO is also applicable to a wide variety of other use cases. For example, it can be used to coordinate the Virtual Private Network (VPN) links of mobile nodes (e.g., cellphones, tablets, laptop computers, etc.) that connect into a home enterprise network via public access networks using services such as OpenVPN [OVPN]. Other applicable use cases are also in scope.

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

2. Terminology

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

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

IPv6 Prefix Delegation (PD)
: a networking service for delegating IPv6 prefixes to nodes on the link. The nominal PD service is DHCPv6 [RFC8415], however alternate services (e.g., based on ND messaging) are also in scope [I-D.templin-v6ops-pdhost][I-D.templin-6man-dhcpv6-ndopt]. Most notably, a minimal form of PD known as "prefix registration" can be used if the Client knows its prefix in advance and can represent it in the IPv6 source address of an ND message.

Access Network (ANET)
: a node's first-hop data link service network, e.g., a radio access network, cellular service provider network, corporate enterprise network, or the public Internet itself. For secured ANETs, link-layer security services such as IEEE 802.1X and physical-layer security prevent unauthorized access internally while border network-layer security services such as firewalls and proxies prevent unauthorized outside access.

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

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

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

INET Partition
frequently, INETs such as large corporate enterprise networks are sub-divided internally into separate isolated partitions. Each partition is fully connected internally but disconnected from other partitions, and there is no requirement that separate partitions maintain consistent Internet Protocol and/or addressing plans. (An INET partition is the same as a SPAN segment discussed below.)

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

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

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

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

underlying interface

an ANET or INET interface over which an AERO interface is configured.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

link-layer address
: an IP address used as an encapsulation header source or destination address from the perspective of the AERO interface.

When an upper layer protocol (e.g., UDP) is used as part of the encapsulation, the port number is also considered as part of the link-layer address. From the perspective of the AERO interface, the link-layer address is either an INET address for intra-segment encapsulation or a SPAN address for inter-segment encapsulation.

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

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

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

Mobile Router (MR)
a MN's on-board router that forwards packets between any downstream-attached networks and the AERO link.

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

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

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

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

Distributed Mobility Management (DMM)

a BGP-based overlay routing service coordinated by Servers and Relays that tracks all Server-to-Client associations.

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

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

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

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

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

3. 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

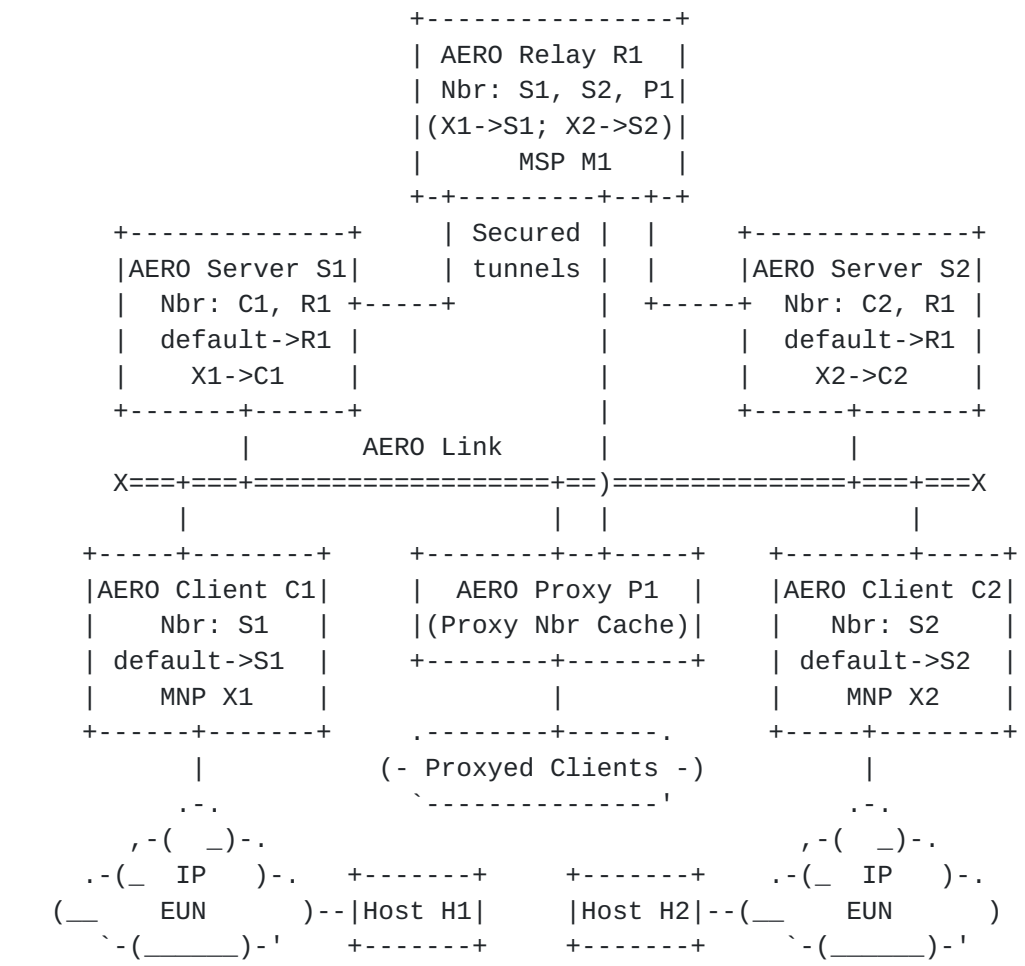

Figure 1: AERO Link Reference Model

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

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

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

- AERO Servers S1 and S2 configure secured tunnels with Relay R1 and also provide mobility, multilink and default router services for their associated Clients C1 and C2.

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

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

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

3.2. AERO Node Types

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

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

AERO Clients register their MNPs through PD exchanges with AERO Servers over the AERO link, and distribute the MNPs to nodes on EUNs. A Client may also be co-resident on the same physical or virtual platform as a Server; in that case, the Client and Server behave as a single functional unit.

AERO Proxys provide a conduit for ANET AERO Clients to associate with AERO Servers in external INETs. Client and Servers exchange control plane messages via the Proxy acting as a bridge between the ANET/INET boundary. The Proxy forwards data packets between Clients and the AERO link according to forwarding information in the neighbor cache. The Proxy function is specified in Section 3.16.

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

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

3.3. AERO Routing System

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

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

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

associated Clients) and they forward all other packets to Relays which have full topology knowledge.

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

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

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

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

Assuming up to 1K sets of Relays, the AERO routing system can then accommodate 1B or more MNPs with no additional overhead (for example, it should be possible to service 1B /64 MNPs taken from a /34 MSP and even more for shorter prefixes). In this way, each set of Relays services a specific set of MSPs that they advertise to the native Internetwork routing system, and each Server configures MSP-specific routes that list the correct set of Relays as next hops. This arrangement also allows for natural incremental deployment, and can support small scale initial deployments followed by dynamic

deployment of additional Clients, Servers and Relays without disturbing the already-deployed base.

Server and Relays can use the Bidirectional Forwarding Detection (BFD) protocol [RFC5880] to quickly detect link failures that don't result in interface state changes, BGP peer failures, and administrative state changes. BFD is important in environments where rapid response to failures is required for routing reconvergence and, hence, communications continuity.

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].

3.3.1. IPv4 Compatibility Routing

For IPv6 MNPs, the AERO routing system includes ordinary IPv6 routes. For IPv4 MNPs, the AERO routing system includes IPv6 routes based on an IPv4-embedded IPv6 address format discussed in Section 3.5.1.

3.4. AERO Addresses

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

For IPv6, Client AERO addresses begin with the prefix fe80::/64 and include in the interface identifier (i.e., the lower 64 bits) the most-significant 64 bits of the Client's IPv6 MNPs. For example, if the AERO Client receives the IPv6 MNP: 2001:db8:1000:2000::/56 it constructs its corresponding AERO address as: fe80::2001:db8:1000:2000.

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

   0:0:0:0:0:FFFF:192.0.2.16/124 (also written as
   0:0:0:0:0:FFFF:c000:0210/124)

The Client then constructs its AERO address 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.

Mobility Service (MS) AERO addresses (used by Relays, Servers, Gateways and Proxys) are allocated from the range fe80::/96, and MUST be managed for uniqueness. The lower 32 bits of the AERO address includes a unique integer value between 1 and 0xfeffffff (e.g., fe80::1, fe80::2, fe80::3, etc., fe80::feff:ffff) as assigned by the administrative authority for the link. If the link spans multiple SPAN segments, the AERO addresses are assigned to each segment in 1x1 correspondence with SPAN addresses (see: Section 3.5). The address fe80:: is the IPv6 link-local Subnet-Router anycast address, and the address fe80::ffff:ffff is the "All-AERO-Servers" address. The address range fe80::ff00:0000/104 is reserved for future use.

The Client's Subnet-Router anycast address can be statelessly determined from its AERO address by simply transposing the AERO address into the upper N bits of the Anycast address followed by 128-N bits of zeroes. For example, for the AERO address fe80::2001:db8:1:2 the Subnet-Router anycast address is 2001:db8:1:2::.

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

3.5. Spanning Partitioned AERO Networks (SPAN)

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

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

for joining the segments therefore is through inter-domain peerings between AERO Relays.

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

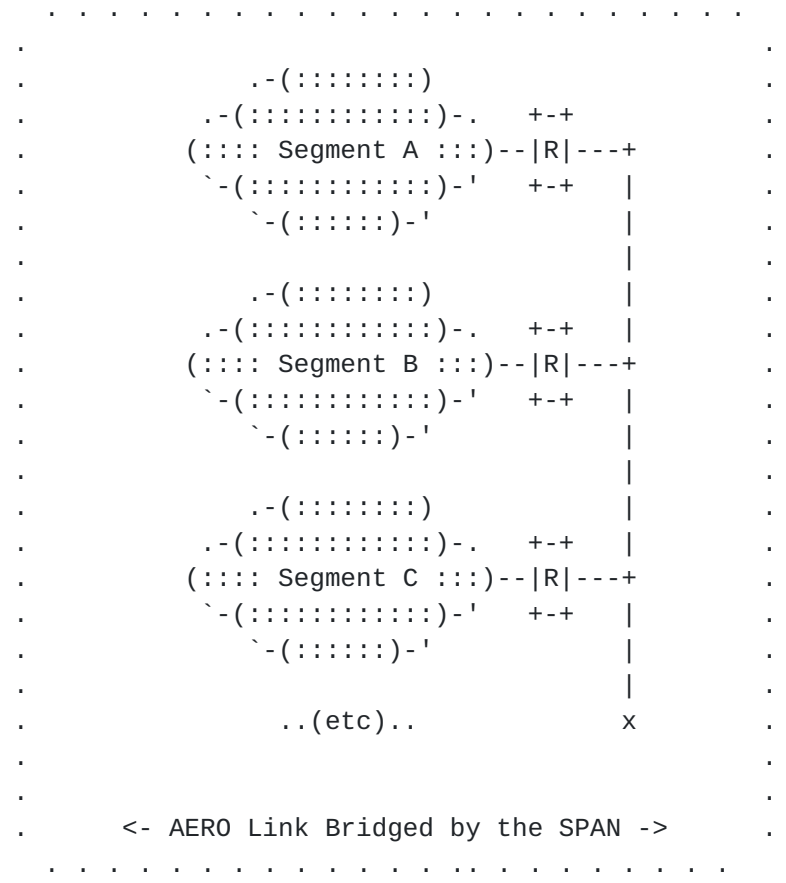

Figure 2: The SPAN

To support the SPAN, AERO links use the Unique Local Address (ULA) prefix fd80::/10 [RFC4193] as the SPAN Service Prefix (SSP). The prefix length intentionally matches the IPv6 link-local prefix (fe80::/10), and enables a simple 1-bit stateless translation between link-local and SPAN prefixes (i.e., bit 7 is '1' for link-local or '0' for SPAN).

Each segment in the SPAN assigns a unique sub-prefix of SSP::/96 termed a "SPAN Partition Prefix (SPP)". For example, a first segment could assign fd80::1000/116, a second could assign fd80::2000/116, a third could assign fd80::3000/116, etc. The administrative

authorities for each segment must therefore coordinate to assure mutually-exclusive SPP assignments, but internal provisioning of the SPP is an independent local consideration for each administrative authority.

An administratively-assigned "SPAN address" is an address taken from a SPP and assigned to a Relay, Server, Gateway or Proxy interface. SPAN addresses are formed by simply clearing bit 7 of an administratively-assigned AERO address. For example, if the SPP is fd80::1000/116, the SPAN address formed from the AERO address fe80::1001 is simply fd80::1001.

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

AERO Relays serve as bridges to join multiple segments into a unified AERO link over multiple diverse administrative domains. They support the bridging function by first establishing forwarding table entries for their SPPs either via standard BGP routing or static routes. For example, if three Relays ('A', 'B' and 'C') from different segments serviced the SPPs fd80::1000/116, fd80::2000/116 and fd80::3000/116 respectively, then the forwarding tables in each Relay are as follows:

A: fd80::1000/116->local, fd80::2000/116->B, fd80::3000/116->C

B: fd80::1000/116->A, fd80::2000/116->local, fd80::3000/116->C

C: fd80::1000/116->A, fd80::2000/116->B, fd80::3000/116->local

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

With the SPPs in place in each Relay's forwarding table, control and data packets sent between AERO nodes in different segments can therefore be carried over the SPAN via encapsulation. For example, when a source AERO node in segment A forwards a packet with IPv6 address 2001:db8:1:2::1 to a target AERO node in segment C with IPv6 address 2001:db8:1000:2000::1, it first encapsulates the packet in a SPAN header with source SPAN address taken from fd80::1000/116 (e.g., fd80::1001) and destination SPAN address taken from fd80::3000/116 (e.g., fd80::3001). Next, it encapsulates the SPAN message in an INET header with source address set to its own INET address (e.g.,

192.0.2.100) and destination set to the INET address of a Relay (e.g., 192.0.2.1).

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

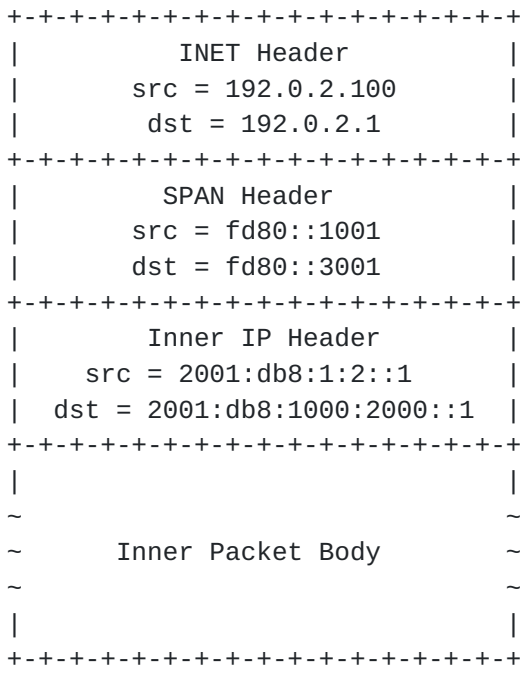

Figure 3: SPAN Encapsulation

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

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

With reference to Figure 3, for a Client's AERO address the SPAN destination address is simply set to the Subnet-Router anycast address. For non-link-local addresses, the destination SPAN address may not be known in advance for the first few packets of a flow sent

via the SPAN. In that case, the SPAN destination address is set to the original packet's destination, and the SPAN routing system will direct the packet to the correct SPAN egress node. (In the above example, the SPAN destination address is simply 2001:db8:1000:2000::1.)

3.5.1. SPAN Compatibility Addressing

For IPv4 MNPs, Servers inject a "SPAN Compatibility Prefix (SCP)" that embeds the MNP into the BGP routing system. The SCP begins with the upper 64 bits of the SSP, followed by the constant string "0000:FFFF" followed by the IPv4 MNP. For example, if the MNP is 192.0.2.0/24 then the SCP is fd80::FFFF:192.0.2.0/120.

This allows for encapsulation of IPv4 packets in IPv6 headers with "SPAN Compatibility Addresses (SCAs)". In this example, the SCA corresponding to the SCP is simply fd80::FFFF:192.0.2.0, and can be used as the SPAN destination address for packets forwarded via the SPAN. This allows for forwarding of initial IPv4 packets over IPv6 SPAN routes, followed by route optimization for direct communications.

3.5.2. Client SPAN Addresses

When an AERO Client or Proxy encapsulates and fragments a packet (see: Section 3.12), it inserts its "Client SPAN Address" as the IPv6 source address of the encapsulation header. This is necessary to provide reassemblers with a source address corresponding to the node that actually inserted the fragment header so that the correct Identification value context is provided.

The Client SPAN address is formed by simply clearing bit 7 of the Client's AERO address. For example, for the Client AERO address fe80::2001:db8:1:2 the corresponding Client SPAN address is fd80::2001:db8:1:2.

Note that the Client's MNP itself (and not the Client SPAN address) is injected into the routing system due to the /64 assumption in the AERO address construction [RFC7421]. Because of the /64 assumption, the most-significant 64 bits of the Client's MNP are written into the least-significant 64 bits of the AERO address. If MNPs longer than /64 are used in the future (i.e., /65 up to /118) the least-significant bits of the MNP would need to be written into bits 10 through 63 of the SPAN address, which would render the address format useless for longest-prefix-match. For more details, see Appendix B of [I-D.templin-6man-omni-interface].

3.6. AERO Interface Characteristics

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

- Native interfaces have global IP addresses that are reachable from any INET correspondent. All Server, Gateway and Relay interfaces are native interfaces, as are INET-facing interfaces of Proxys.

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

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

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

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

AERO interfaces use encapsulation (see: Section 3.9) to exchange packets with AERO link neighbors over Native, NATed or VPNed interfaces. AERO interfaces do not use encapsulation over Proxyed and Direct underlying interfaces.

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

AERO interfaces send ND messages with an Overlay Multilink Network Interface (OMNI) option formatted as specified in [I-D.templin-6man-omni-interface]. The OMNI option includes prefix

registration information and "ifIndex-tuples" containing link quality information for the AERO interface's underlying interfaces.

When encapsulation is used, AERO interface ND messages MAY also include an AERO Source/Target Link-Layer Address Option (S/TLLAO) formatted as shown in Figure 4:

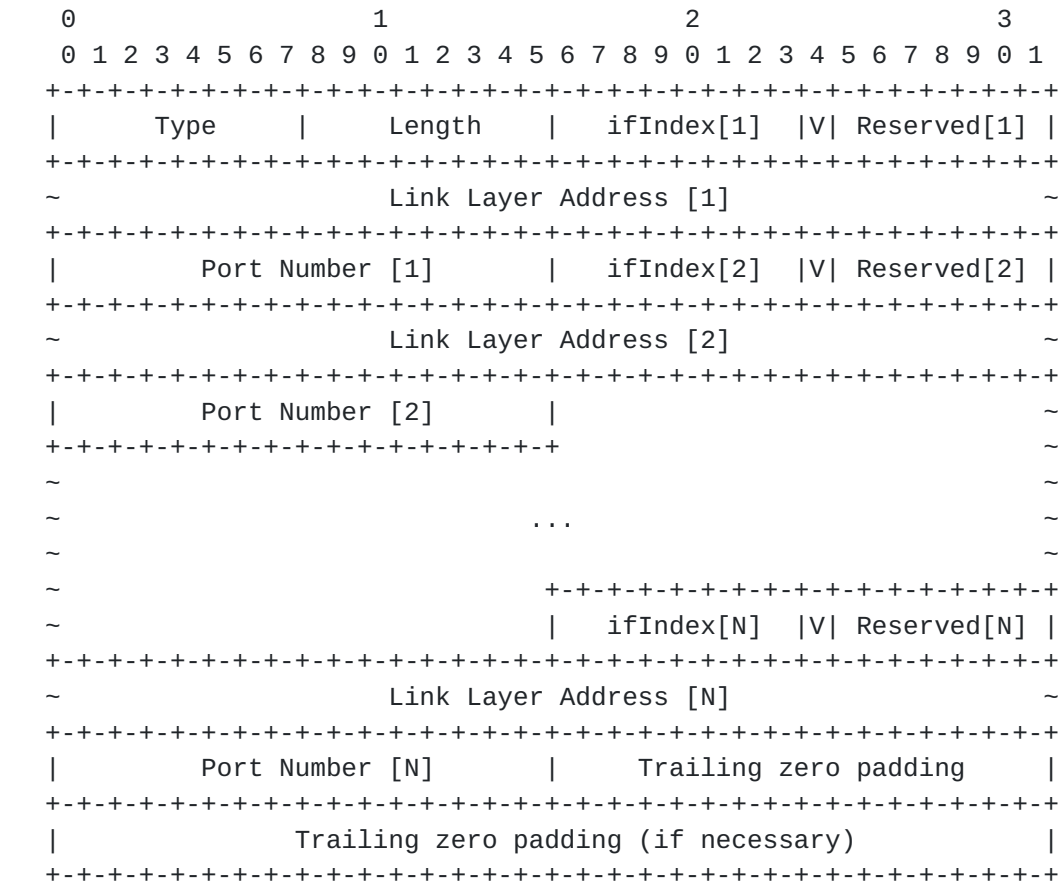

Figure 4: AERO Source/Target Link-Layer Address Option (S/TLLAO) Format

In this format, Type and Length are set the same as specified for S/TLLAOs in [RFC4861], with trailing zero padding octets added as necessary to produce an integral number of 8 octet blocks. The S/TLLAO includes N ifIndex-tuples in correspondence to ifIndex-tuples that appear in the OMNI option. Each ifIndex-tuple includes the folllowing information:

- ifIndex[i] - the same value as in the corresponding ifIndex-tuple included in the OMNI option.

- V[i] - a bit that identifies the IP protocol version of the address found in the Link Layer Address [i] field. The bit is set to 0 for IPv4 or 1 for IPv6.

- Reserved[i] - MUST encode the value 0 on transmission, and ignored on reception.

- Link Layer Address [i] - the IPv4 or IPv6 address used as the encapsulation source address. The field is 4 bytes in length for IPv4 or 16 bytes in length for IPv6.

- Port Number [i] - the upper layer protocol port number used as the encapsulation source port, or 0 when no upper layer protocol encapsulation is used. The field is 2 bytes in length.

If an S/TLLAO is included, any ifIndex-tuples correspond to a proper subset of the OMNI option ifIndex-tuples. Any S/TLLAO ifIndex-tuple having an ifIndex value that does not appear in an OMNI option ifindex-tuple is ignored. If the same ifIndex value appears in multiple ifIndex-tuples, the first tuple is processed and the remaining tuples are ignored. Any S/TLLAO ifIndex-tuples can therefore be viewed as inter-dependent extensions of their corresponidng OMNI option ifIndex-tuples, i.e., the OMNI option and S/TLLAO are companions that are interpreted in conjunction with each other.

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

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

If the Client has multiple active underlying interfaces, then from the perspective of ND it would appear to have multiple link-layer addresses. In that case, ND message OMNI options MAY include multiple ifIndex-tuples - each with a value that corresponds to a specific interface. Every ND message need not include all OMNI and/

or S/TLLAO ifIndex-tuples; for any ifIndex-tuple not included, the neighbor considers the status as unchanged.

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

3.7. AERO Interface Initialization

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

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

3.7.1. AERO Server/Gateway Behavior

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

The AERO interface provides a single interface abstraction to the IP layer, but internally comprises multiple secured tunnels as well as an NBMA nexus for sending encapsulated data packets to AERO interface neighbors. The Server further configures a service to facilitate ND/

PD exchanges with AERO Clients and manages per-Client neighbor cache entries and IP forwarding table entries based on control message exchanges.

Gateways are simply Servers that run a dynamic routing protocol between the AERO interface and INET/EUN interfaces (see: Section 3.3). The Gateway provisions MNPs to networks on the INET/EUN interfaces (i.e., the same as a Client would do) and advertises the MSP(s) for the AERO link over the INET/EUN interfaces. The Gateway further provides an attachment point of the AERO link to the non-MNP-based global topology.

3.7.2. AERO Proxy Behavior

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

3.7.3. AERO Client Behavior

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

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

3.7.4. AERO Relay Behavior

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

3.8. AERO Interface Neighbor Cache Maintenance

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

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

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

Asymmetric neighbor cache entries are created or updated based on route optimization messaging as specified in Section 3.17, and are garbage-collected when keepalive timers expire. AERO route optimization sources (ROSs) maintain asymmetric neighbor cache entries for active targets with lifetimes based on ND messaging constants. Asymmetric neighbor cache entries are unidirectional since only the ROS (and not the target) creates an entry.

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

To the list of neighbor cache entry states in Section 7.3.2 of [RFC4861], Proxy and Server AERO interfaces add an additional state DEPARTED that applies to symmetric and proxy neighbor cache entries for Clients that have recently departed. The interface sets a "DepartTime" variable for the neighbor cache entry to "DEPARTTIME" seconds. DepartTime is decremented unless a new ND message causes the state to return to REACHABLE. While a neighbor cache entry is in the DEPARTED state, packets destined to the target Client are forwarded to the Client's new location instead of being dropped.

When DepartTime decrements to 0, the neighbor cache entry is deleted. It is RECOMMENDED that DEPARTTIME be set to the default constant value REACHABLETIME plus 10 seconds (40 seconds by default) to allow a window for packets in flight to be delivered while stale route optimization state may be present.

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

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

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

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

3.9. AERO Interface Encapsulation and Re-encapsulation

Client AERO interfaces avoid encapsulation over Direct underlying interfaces and Proxyed underlying interfaces for which the first-hop access router is AERO-aware. Other AERO interfaces encapsulate packets according to whether they are entering the AERO interface from the network layer or if they are being re-admitted into the same AERO link they arrived on. This latter form of encapsulation is known as "re-encapsulation".

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

For packets undergoing re-encapsulation, the AERO interface instead copies these values from the original encapsulation header into the new encapsulation header, i.e., the values are transferred between encapsulation headers and *not* copied from the encapsulated packet's network-layer header. (Note especially that by copying the TTL/Hop Limit between encapsulation headers the value will eventually decrement to 0 if there is a (temporary) routing loop.) For IPv4 encapsulation/re-encapsulation, the AERO interface sets the DF bit as discussed in Section 3.12.

AERO interfaces configured over INET underlying interfaces encapsulate each packet in a SPAN header, then encapsulate the resulting SPAN packet in an INET header according to the next hop determined in the forwarding algorithm in Section 3.13. If the next hop is reached via a secured tunnel, the AERO interface uses an INET encapsulation format specific to the secured tunnel type (see: Section 6). If the next hop is reached via an unsecured underlying interface, the AERO interface instead uses Generic UDP Encapsulation (GUE) [I-D.ietf-intarea-gue] or an alternate minimal encapsulation format Appendix A.

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

AERO interfaces observes the packet sizing and fragmentation considerations found in Section 3.12.

3.10. AERO Interface Decapsulation

AERO interfaces decapsulate packets destined either to the AERO node itself or to a destination reached via an interface other than the AERO interface the packet was received on. When the encapsulated packet arrives in multiple fragments, the AERO interface reassembles as discussed in Section 3.12. Further decapsulation steps are performed according to the appropriate encapsulation format specification.

3.11. AERO Interface Data Origin Authentication

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

- AERO Relays, Servers and Proxys accept encapsulated data packets and control messages received from secured tunnels via the SPAN.
- AERO Servers and Proxys accept encapsulated data packets and NS messages used for Neighbor Unreachability Detection (NUD) received from a member of the ROS list.
- AERO Proxys and Clients accept packets that originate from within the same secured ANET.
- AERO Clients and Gateways accept packets from downstream network correspondents based on ingress filtering.

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

3.12. AERO Interface MTU and Fragmentation

All IPv6 interfaces are REQUIRED to configure a minimum Maximum Transmission Unit (MTU) of 1280 bytes [RFC8200]. (IPv4 interfaces have a smaller minimum MTU [RFC1122], but SHOULD also observe the IPv6 minimum MTU if possible.) The AERO link therefore MUST forward IPv6 packets of at least 1280 bytes without generating an IPv6 Path MTU Discovery (PMTUD) Packet Too Big (PTB) message [RFC8201].

The AERO interface configures an MTU of 9180 bytes [RFC2492]; the size is therefore not a reflection of the underlying interface MTUs, but rather determines the largest packet the AERO interface can forward or reassemble.

The AERO interface can employ link-layer IPv6 encapsulation and fragmentation/reassembly per [RFC2473], but its use is OPTIONAL since correct operation will result in either case. Implementations that omit link-layer IPv6 fragmentation/reassembly may be more prone to dropping large packets and returning a PTB, while those that include it may see improved performance at the expense of including additional code. In both cases, AERO interface neighbors are responsible for advertising their willingness to reassemble.

The AERO interface returns internally-generated PTB messages for packets admitted into the interface that it deems too large for the outbound underlying interface (e.g., according to underlying interface performance characteristics, MTU, etc). For all other packets, the AERO interface performs PMTUD even if the destination appears to be on the same link since a proxy on the path could return a PTB message. This ensures that the path MTU is adaptive and reflects the current path used for a given data flow.

When a Client's AERO interface sends a packet that is no larger than the MTU of the selected underlying interface, it sends according to the underlying interface L2 frame format. When the AERO interface sends a packet that is larger than the underlying interface MTU, it drops the packet and returns a PTB if the neighbor is not willing to reassemble.

Otherwise, the AERO interface encapsulates the packet in an IPv6 header per [RFC2473] with source address set to the Client's link-local address and destination address set to the link-local address of the next hop. The AERO interface then uses IPv6 fragmentation to break the encapsulated packet into fragments that are no larger than the underlying interface MTU and sends the fragments over the underlying interface. The next hop then reassembles and conveys the packet toward the final destination.

When a Proxy or Server receives a fragmented or whole packet from the INET destined to a Client, it must determine whether to forward or drop and return a PTB (e.g., according to the underlying interface performance characteristics, MTU, etc). If the Proxy/Server deems the packet to be of acceptable size, it first reassembles locally (if necessary) then forwards the packet to the Client. If the (reassembled) packet is no larger than the underlying interface MTU, the Proxy/Server forwards according to the underlying interface L2 frame format. If the packet is larger than the MTU, the Proxy/Server instead uses link-layer encapsulation and IPv6 fragmentation as above if the Client accepts fragments or drops and returns a PTB otherwise. The Client then reassembles and discards the encapsulation header, then forwards the whole packet to the final destination.

When a Proxy, Server or Gateway forwards a Client's packet over the SPAN, it uses IPv6 encapsulation with its own SPAN address as the source address and the SPAN address of the next hop as the destination, then uses fragmentation to break the SPAN-encapsulated packet into pieces no larger than 1280 bytes. When a Server or Gateway forwards a SPAN-encapsulated packet to a destination outside of the AERO link, it first reassembles if necessary. This implies that Proxys, Servers and Gateways MUST support fragmentation and reassembly for packet exchanges over the SPAN even though fragmentation and reassembly is OPTIONAL for Clients.

Applications that cannot tolerate loss due to MTU restrictions SHOULD avoid sending packets larger than 1280 bytes, since dynamic path changes can reduce the path MTU at any time. Applications that may benefit from sending larger packets even though the path MTU may change dynamically MAY use larger sizes (i.e., up to the AERO interface MTU).

Note that when a Proxy/Server forwards a fragmented packet received from the INET to a Client, it reassembles locally first instead of blindly forwarding fragments directly to the Client to avoid attacks such as tiny fragments, overlapping fragments, etc.

Note also that the AERO interface can forward large packets via encapsulation and fragmentation while at the same time returning advisory PTB messages, e.g., subject to rate limiting. The interface can therefore continuously forward large packets without loss while sending advisory messages recommending a smaller size. Even more appropriately, the receiving OMNI node that performs reassembly can send advisory PTB messages if reassembly conditions are currently unfavorable.

3.13. AERO Interface Forwarding Algorithm

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

Packets that enter the AERO interface from the network layer are forwarded to an AERO interface neighbor. Packets that enter the AERO interface from the link layer are either re-admitted into the AERO link or forwarded to the network layer where they are subject to either local delivery or IP forwarding. In all cases, the AERO

interface itself MUST NOT decrement the network layer TTL/Hop-count since its forwarding actions occur below the network layer.

AERO interfaces may have multiple underlying interfaces and/or neighbor cache entries for neighbors with multiple ifIndex-tuple registrations (see Section 3.6). The AERO interface uses each packet's DSCP value (and/or other traffic discriminators such as port number) 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 replicates the packet and sends one copy via each of the (outgoing / neighbor) interface pairs; otherwise, the node sends a single copy of the packet via an 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, Proxys, Servers and Relays. In the following discussion, a packet's destination address is said to "match" if it is the same as a cached address, or if it is covered by a cached prefix (which may be encoded in an AERO address).

3.13.1. Client Forwarding Algorithm

When an IP packet enters a Client's AERO interface from the network layer the Client searches for an asymmetric neighbor cache entry that matches the destination. If there is a match, the Client uses one or more "reachable" neighbor interfaces in the entry for packet forwarding. If there is no asymmetric neighbor cache entry, the Client instead forwards the packet toward a Server (the packet is intercepted by a Proxy if there is a Proxy on the path). The Client encapuslates the packet in an IPv6 header and fragments if necessary according to MTU requirements (see: Section 3.12).

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

3.13.2. Proxy Forwarding Algorithm

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

When the Proxy receives a data packet from a Client within the ANET, it first reassembles and decapsulates if the packet was a link-local fragment. The Proxy next inserts a SPAN header with source address set to the Proxy's SPAN address and destination address set to the SPAN address of the next hop. The Proxy then fragments the SPAN packet if necessary into fragments no larger than 1280 bytes, then searches for an asymmetric neighbor cache entry that matches the destination and forwards the fragments as follows:

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

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

When the Proxy receives an encapsulated data packet from an INET neighbor or from a secured tunnel from a Relay, it accepts the packet only if data origin authentication succeeds and if there is a proxy neighbor cache entry that matches the inner destination. Next, if the packet is a SPAN fragment the Proxy adds the fragment to the reassembly buffer. The Proxy then reassembles the packet (if necessary) and continues processing.

Next if reassembly is complete and the neighbor cache state is REACHABLE, the Proxy either drops and returns a PTB (see: Section 3.12) or forwards the packet to the Client while performing link-local encapsulation and re-fragmentation to the ANET MTU size if necessary. If the neighbor cache entry state is DEPARTED, the Proxy instead changes the SPAN destination address to the address of the new Server and forwards it to a Relay while performing re-fragmentation to 1280 bytes if necessary.

When the Proxy forwards a SPAN packet to a REACHABLE Client for which the packet is no larger than the ANET MTU, it decapsulates the SPAN header first and forwards the (unencapsulated) packet to the Client to avoid the unnecessary overhead for carrying the SPAN header.

3.13.3. Server/Gateway Forwarding Algorithm

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

When the Server receives an encapsulated data packet from an INET neighbor or from a secured tunnel, it accepts the packet only if data origin authentication succeeds. If the SPAN destination address is its own address, the Server continues processing as follows:

- if the destination matches a symmetric neighbor cache entry in the REACHABLE state the Server prepares the packet for forwarding to the destination Client. For the Client's Proxyed interfaces, the Server changes the SPAN destination address to the address of the Proxy and forwards the packet to the Proxy. For the Client's other interfaces, the Server reassembles then either drops and returns a PTB (see: Section 3.12) or forwards the packet (while re-fragmenting if necessary) using SPAN encapsulation for the Client's Native, NATed or VPNed interfaces, or no encapsulation for Direct interfaces.

- else, if the destination matches a symmetric neighbor cache entry in the DEPARETED state the Server re-encapsulates the packet and forwards it using the SPAN address of the Client's new Server as the destination.

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

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

- else, the Server (acting as a Gateway) reassembles if necessary, decapsulates the packet and releases it to the network layer for local delivery or IP forwarding. Based on the information in the

forwarding table, the network layer may return the packet to the same AERO interface in which case further processing occurs as below. (Note that this arrangement accommodates common implementations in which the IP forwarding table is not accessible from within the AERO interface. If the AERO interface can directly access the IP forwarding table (such as for in-kernel implementations) the forwarding table lookup can instead be performed internally from within the AERO interface itself.)

When the Server's AERO interface receives a data packet from the network layer or from a NATed/VPNed/Direct Client, it performs SPAN encapsualtion and fragmentation if necessary, then processes the packet according to the network-layer destination address as follows:

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

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

3.13.4. Relay Forwarding Algorithm

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

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

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

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

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

3.14. AERO Interface Error Handling

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

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

The IP header is followed by an ICMP header that includes an error Type, Code and Checksum. Valid type values include "Destination Unreachable", "Time Exceeded" and "Parameter Problem" [RFC0792][RFC4443]. (AERO interfaces ignore all link-layer IPv4 "Fragmentation Needed" and IPv6 "Packet Too Big" messages since they only emit packets that are guaranteed to be no larger than the IP minimum link MTU as discussed in Section 3.12.)

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 5 (where, "L2" and "L3" refer to link-layer and network-layer, respectively):

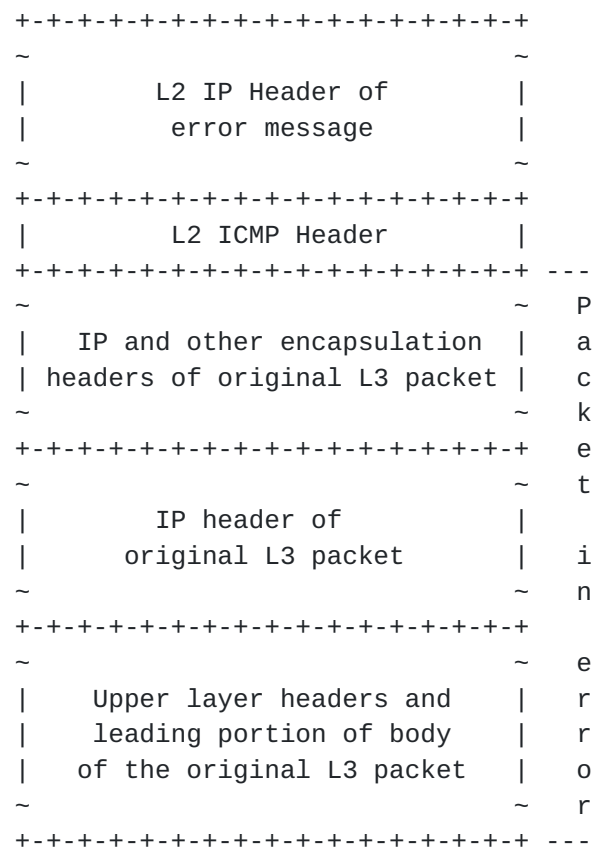

Figure 5: AERO Interface Link-Layer Error Message Format

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

- 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].

- 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.

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

- When an AERO Client receives persistent link-layer Destination Unreachable messages in response to encapsulated packets that it sends to one of its symmetric neighbor Servers, the Client should mark the path as unusable and use another path. If it receives Destination Unreachable messages on many or all paths, the Client should associate with a new Server and release its association with the old Server as specified in Section 3.19.5.

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

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

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

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

3.15. AERO Router Discovery, Prefix Delegation and Autoconfiguration

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

3.15.1. AERO ND/PD Service Model

Each AERO Server on the link configures a PD service to facilitate Client requests. Each Server is provisioned with a database of MNP-to-Client ID mappings for all Clients enrolled in the AERO service,

as well as any information necessary to authenticate each Client. The Client database is maintained by a central administrative authority for the AERO link and securely distributed to all Servers, e.g., via the Lightweight Directory Access Protocol (LDAP) [RFC4511], via static configuration, etc. Clients receive the same service regardless of the Servers they select.

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

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

The following sections specify the Client and Server behavior.

3.15.2. AERO Client Behavior

AERO Clients discover the addresses of Servers in a similar manner as described in [RFC5214]. Discovery methods include 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]. Alternatively, the Client can discover Server addresses through a layer 2 data link login exchange, or through a unicast RA response to a multicast/anycast RS as described below. In the absence of other information, the Client can resolve the DNS Fully-Qualified Domain Name (FQDN) "linkupnetworks.[domainname]" where "linkupnetworks" is a constant text string and "[domainname]" is a DNS suffix for the AERO link (e.g., "example.com").

To associate with a Server, the Client acts as a requesting router to request MNPs. The Client prepares an RS message with PD parameters and includes a Nonce and Timestamp option if the Client needs to correlate RA replies. If the Client already knows the Server's AERO address, it includes the AERO address as the network-layer destination address; otherwise, it includes the link-scoped All-

Routers multicast (ff02::2) or Subnet-Router anycast (fe80::) address as the network-layer destination. If the Client already knows its own AERO address, it uses the AERO address as the network-layer source address; otherwise, it uses the unspecified IPv6 address (::/128) as the network-layer source address.

The Client next includes an OMNI option in the RS message to register its link-layer information with the Server. The Client sets the OMNI option prefix registration information according to the MNP, and includes an ifIndex-tuple with S set to '1' corresponding to the underlying interface over which the Client will send the RS message. The Client MAY include additional ifIndex-tuples specific to other underlying interfaces. The Client MAY also include an SLLAO with a link-layer address corresponding to the OMNI option ifIndex-tuple with S set to '1'.

The Client then sends the RS message (either directly via Direct interfaces, via INET encapsulation for NATed interfaces, via a VPN for VPNed interfaces, via a Proxy for proxyed interfaces or via a Relay for native interfaces) and waits for an RA message reply (see Section 3.15.3). The Client retries up to MAX_RTR_SOLICITATIONS times until an RA is received. If the Client receives no RAs, or if it receives an RA with Router Lifetime set to 0, the Client SHOULD abandon this Server and try another Server. Otherwise, the Client processes the PD information found in the RA message.

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

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

Following autoconfiguration, the Client sub-delegates the MNPs to its attached EUNs and/or the Client's own internal virtual interfaces as

described in [I-D.templin-v6ops-pdhost] to support the Client's downstream attached "Internet of Things (IoT)". The Client subsequently sends additional RS messages over each underlying interface before the Router Lifetime received for that interface expires.

After the Client registers its underlying interfaces, it may wish to change one or more registrations, e.g., if an interface changes address or becomes unavailable, if QoS preferences change, etc. To do so, the Client prepares an RS message to send over any available underlying interface. The RS includes an OMNI option with prefix registration information specific to its MNP, with an ifIndex-tuple specific to the selected underlying interface with S set to '1', and with any additional ifIndex-tuples specific to other underlying interfaces. The Client includes fresh ifIndex-tuple values to update the Server's neighbor cache entry. When the Client receives the Server's RA response, it has assurance that the Server has been updated with the new information.

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

3.15.3. AERO Server Behavior

AERO Servers act as IP routers and support a PD service for Clients. Servers arrange to add their AERO 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. Server addresses should be geographically and/or topologically referenced, and made available for discovery by Clients on the AERO link.

When a Server receives a prospective Client's RS message on its AERO interface, it SHOULD return an immediate RA reply with Router Lifetime set to 0 if it is currently too busy or otherwise unable to service the Client. Otherwise, the Server authenticates the RS message and processes the PD parameters. The Server first determines the correct MNPs to delegate to the Client by searching the Client database. When the Server delegates the MNPs, it also creates a forwarding table entry for each MNP so that the MNPs are propagated into the routing system (see: Section 3.3). For IPv6, the Server creates an IPv6 forwarding table entry for each MNP. For IPv4, the Server creates an IPv6 forwarding table entry with the SPAN Compatibility Prefix (SCP) corresponding to the IPv4 address.

The Server next creates a symmetric neighbor cache entry for the Client using the base AERO address as the network-layer address and with lifetime set to no more than the smallest PD lifetime. Next, the Server updates the neighbor cache entry by recording the information in each ifIndex-tuple in the RS OMNI option. The Server also records the actual SPAN/INET addresses in the neighbor cache entry. If an SLLAO was present, the Server also compares the SLLAO address information for the first ifIndex-tuple with the SPAN/INET information to determine if there is a NAT on the path.

Next, the Server prepares an RA message using its AERO address as the network-layer source address and the network-layer source address of the RS message as the network-layer destination address. The Server sets the Router Lifetime to the time for which it will maintain both this underlying interface individually and the symmetric neighbor cache entry as a whole. The Server also sets Cur Hop Limit, M and O flags, Reachable Time and Retrans Timer to values appropriate for the AERO link. The Server includes the delegated MNPs, any other PD parameters and an OMNI option with no ifIndex-tuples. The Server then includes one or more RIOs that encode the MSPs for the AERO link, plus an MTU option (see Section 3.12). The Server finally forwards the message to the Client using SPAN/INET, INET, or NULL encapsulation as necessary.

After the initial RS/RA exchange, the Server maintains a ReachableTime timer for each of the Client's underlying interfaces individually (and for the Client's symmetric neighbor cache entry collectively) set to expire after Router Lifetime seconds. If the Client (or Proxy) issues additional RS messages, the Server sends an RA response and resets ReachableTime. If the Server receives an ND message with PD release indication it sets the Client's symmetric neighbor cache entry to the DEPARTED state and withdraws the MNP from the routing system after a short delay (e.g., 2 seconds). If ReachableTime expires before a new RS is received on an individual underlying interface, the Server marks the interface as DOWN. If ReachableTime expires before any new RS is received on any individual underlying interface, the Server deletes the neighbor cache entry and withdraws the MNP without delay.

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

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

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

3.15.3.1. Lightweight DHCPv6 Relay Agent (LDRA)

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

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

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

When the DHCPv6 server prepares a Reply message, it wraps the message in a 'Relay-Reply' message and echoes the Interface-Id option. The DHCPv6 server then delivers the Relay-Reply message to the LDRA, which discards the Relay-Reply wrapper and IPv6/UDP headers, then uses the DHCPv6 message to construct an RA response to the Client. The Server uses the information in the Interface-Id option to prepare

the RA message and to cache the link-layer addresses taken from the OMNI option and SLLAO echoed in the Interface-Id option.

3.16. The AERO Proxy

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

- when the Proxy receives an RS message from a new ANET Client, it first authenticates the message then examines the network-layer destination address. If the destination address is a Server's AERO address, the Proxy proceeds to the next step. Otherwise, if the destination is All-Routers multicast or Subnet-Router anycast, the Proxy selects a "nearby" Server that is likely to be a good candidate to serve the Client and replaces the destination address with the Server's AERO address. Next, the Proxy creates a proxy neighbor cache entry and caches the Client and Server link-layer addresses along with the OMNI option information and any other identifying information including Transaction IDs, Client Identifiers, Nonce values, etc. The Proxy finally encapsulates the (proxyed) RS message in a SPAN header with source set to the Proxy's SPAN address and destination set to the Server's SPAN address then forwards the message into the SPAN.

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

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

After the initial RS/RA exchange, the Proxy forwards any Client data packets for which there is no matching asymmetric neighbor cache entry to a Relay via the SPAN. The Proxy instead forwards any Client data destined to an asymmetric neighbor cache target directly to the

target according to the link-layer information - the process of establishing asymmetric neighbor cache entries is specified in Section 3.17.

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

If the Server ceases to send solicited advertisements, the Proxy sends unsolicited RAs on the ANET interface with destination set to All-Nodes multicast (ff02::1) and with Router Lifetime set to zero to inform Clients that the Server has failed. Although the Proxy engages in ND exchanges on behalf of the Client, the Client can also send ND messages on its own behalf, e.g., if it is in a better position than the Proxy to convey QoS changes, etc. For this reason, the Proxy marks any Client-originated solicitation messages (e.g. by inserting a Nonce option) so that it can return the solicited advertisement to the Client instead of processsing it locally.

If the Client becomes unreachable, the Proxy sets the neighbor cache entry state to DEPARTED and retains the entry for DEPARTTIME seconds. While the state is DEPARTED, the Proxy forwards any packets destined to the Client to a Relay. The Relay in turn forwards the packets to the Client's current Server. When DepartTime expires, the Proxy deletes the neighbor cache entry and discards any further packets destined to this (now forgotten) Client.

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

If the first-hop ANET access router is AERO-aware, the Client can avoid encapsulation for both its control and data messages. When the Client connects to the link, it can send an unencapsulated RS message with source address set to its AERO address and with destination address set to the AERO address of the Client's selected Server or to All-Routers multicast or Subnet-Router anycast. The Client includes an OMNI option formatted as specified in [I-D.templin-6man-omni-interface].

The Client then sends the unencapsulated RS message, which will be intercepted by the AERO-Aware access router. The access router then encapsulates the RS message in an ANET header with its own address as the source address and the address of a Proxy as the destination address. The access router further remembers the address of the Proxy so that it can encapsulate future data packets from the Client via the same Proxy. If the access router needs to change to a new Proxy, it simply sends another RS message toward the Server via the new Proxy on behalf of the Client.

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

3.16.1. Detecting and Responding to Server Failures

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

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

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

3.16.2. Point-to-Multipoint Server Coordindation

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

When the Proxy receives the RS and processes the OMNI option, it performs a separate RS/RA exchange with each MS-Register Server. When it has received the RA messages, it creates an "aggregate" RA message to return to the Client with an OMNI option with each responding Server's ID recorded in an MS-Register sub-option.

Client's can thereafter employ efficient point-to-multipoint Server coordination under the assistance of the Proxy to dramatically reduce the number of messages sent over the ANET while enlisting the support of multiple Servers for fault tolerance. Clients can further include MS-Release suboptions in RS messages to request the Proxy to release from former Servers via the procedures discussed in Section 3.19.5.

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

3.17. AERO Route Optimization

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

- For Clients on VPNed, NATed and Direct interfaces, the Server is the ROS.
- For Clients on Proxyed interfaces, the Proxy is the ROS.
- For Clients on native interfaces, the Client itself is the ROS.
- For correspondent nodes on INET/EUN interfaces serviced by a Gateway, the Gateway is the ROS.

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

The procedures are specified in the following sections.

3.17.1. Route Optimization Initiation

While data packets are flowing from the source node toward a target node, the ROS performs address resolution by sending an NS message for Address Resolution (NS(AR)) to receive a solicited NA message from the ROR. When the ROS sends an NS(AR), it includes:

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

The NS(AR) message includes an OMNI option with no ifIndex-tuples and no SLLAO, such that the target will not create a neighbor cache entry.

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

3.17.2. Relaying the NS

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

3.17.3. Processing the NS and Sending the NA

When the ROR receives the NS(AR) message, it examines the Target Address to determine whether it has a neighbor cache entry and/or route that matches the target. If there is no match, the ROR drops the NS(AR) message. Otherwise, the ROR continues processing as follows:

- if the target belongs to an MNP Client neighbor in the DEPARTED state the ROR changes the NS(AR) message SPAN destination address

to the SPAN address of the Client's new Server, forwards the message into the SPAN and returns from processing.

- If the target belongs to an MNP Client neighbor in the REACHABLE state, the ROR instead adds the AERO source address to the target Client's Report List with time set to ReportTime.

- If the target belongs to a non-MNP route, the ROR continues processing without adding an entry to the Report List.

The ROR then prepares a solicited NA message to send back to the ROS but does not create a neighbor cache entry. The ROR sets the NA source address to the AERO address corresponding to the target, sets the Target Addresss to the target of the solicitation, and sets the destination address to the source of the solicitation.

The ROR then includes an OMNI option with prefix registration length set to the length of the MNP if the target is an MNP Client; otherwise, set to the maximum of the non-MNP prefix length and 64. (Note that a /64 limit is imposed to avoid causing the ROS to set short prefixes (e.g., "default") that would match destinations for which the routing system includes more-specific prefixes.)

If the target is an MNP Client, the ROR next includes ifIndex-tuples in the OMNI option for each of the target Client's underlying interfaces with current information for each interface and with the S flag set to 0. The ROR then includes a TLLAO with ifIndex-tuples in one-to-one correspondence with the tuples that appear in the OMNI option. For NATed, VPNed and Direct interfaces, the link layer addresses are the SPAN address of the ROR. For Proxyed interfaces, the link-layer addresses are the SPAN addresses of the Proxy's INET interfaces. For native interfaces, the link-layer addresses are the SPAN addesses of the Client's native interfaces.

The ROR then sets the NA message R flag to 1 (as a router), S flag to 1 (as a response to a solicitation), and O flag to 0 (as a proxy). The ROR finally encapsulates the NA message in a SPAN header with source set to its own SPAN address and destination set to the source SPAN address of the NS(AR) message, then forwards the message into the SPAN without decrementing the network-layer TTL/Hop Limit field.

3.17.4. Relaying the NA

When the Relay receives the NA message from the ROR, it discards the INET header and determines that the ROS is the next hop by consulting its standard IPv6 forwarding table for the SPAN header destination address. The Relay then forwards the SPAN-encapsulated NA message

toward the ROS the same as for any IPv6 router. The final-hop Relay in the SPAN will deliver the message via a secured tunnel to the ROS.

3.17.5. Processing the NA

When the ROS receives the solicited NA message, it processes the message the same as for standard IPv6 Address Resolution [RFC4861]. In the process, it caches the source SPAN address then creates an asymmetric neighbor cache entry for the ROR and caches all information found in the OMNI and TLLAO options. The ROS finally sets the asymmetric neighbor cache entry lifetime to REACHABLETIME seconds.

3.17.6. Route Optimization Maintenance

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

While new data packets destined to the target are flowing through the ROS, it sends additional NS(AR) messages to the ROR before ReachableTime expires to receive a fresh solicited NA message the same as described in the previous sections (route optimization refreshment strategies are an implementation matter, with a non-normative example given in Appendix B.1). The ROS uses the cached SPAN address of the ROR as the NS(AR) SPAN destination address, and sends up to MAX_MULTICAST_SOLICIT NS(AR) messages separated by 1 second until an NA is received. If no NA is received, the ROS assumes that the current ROR has become unreachable and deletes the neighbor cache entry. Subsequent data packets will trigger a new route optimization per Section 3.17.1 to discover a new ROR while initial data packets travel over a suboptimal route.

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

The ROS may also receive unsolicited NA messages from the ROR at any time (see: Section 3.19). If there is an asymmetric neighbor cache entry for the target, the ROS updates the link-layer information but does not update ReachableTime since the receipt of an unsolicited NA

does not confirm that any forward paths are working. If there is no asymmetric neighbor cache entry, the ROS simply discards the unsolicited NA.

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

3.18. Neighbor Unreachability Detection (NUD)

AERO nodes perform Neighbor Unreachability Detection (NUD) per [RFC4861] either reactively in response to persistent link-layer errors (see Section 3.14) or proactively to confirm reachability. The NUD algorithm is based on periodic control message exchanges. The algorithm may further be seeded by ND hints of forward progress, but care must be taken to avoid inferring reachability based on spoofed information. For example, authentic IPv6 ND message exchanges may be considered as acceptable hints of forward progress, while spurious data packets should not be.

AERO Servers, Proxys and Gateways can use standard NS/NA NUD exchanges sent over the SPAN to securely test reachability without risk of DoS attacks from nodes pretending to be a neighbor; Proxys can further perform NUD to securely verify Server reachability on behalf of their proxyed Clients. However, a means for a ROS to test the unsecured forward directions of target route optimized paths is also necessary. The following paragraphs present the suggested method.

When an ROR directs an ROS to a neighbor with one or more target link-layer addresses, the ROS can proactively test each such unsecured route optimized path by sending "loopback" NS(NUD) messages. While testing the paths, the ROS can optionally continue to send packets via the SPAN, maintain a small queue of packets until target reachability is confirmed, or (optimistically) allow packets to flow via the route optimized paths.

When the ROS sends a loopback NS(NUD) message, it uses its AERO address as both the IPv6 source and destination address, and any IPv6 address as the Target Address. The ROS includes a Nonce and Timestamp option, then encapsulates the message in SPAN/INET headers with its own SPAN address as the source and the SPAN address of the route optimization target as the destination. The ROS then forwards

the message to the target (either directly to the link layer address of the target if the target is in the same SPAN segment, or via a Relay if the target is in a different SPAN segment).

When the route optimization target receives the NS(NUD) message, it notices that the IPv6 destination address is the same as the source address. It then reverses the SPAN source and destination addresses and returns the message to the ROS (either directly or via the SPAN). The route optimization target does not decrement the NS(NUD) message IPv6 Hop-Limit in the process, since the message has not exited the SPAN.

When the ROS receives the NS(NUD) message, it can determine from the Nonce, Timestamp and Target Address that the message originated from itself and that it transited the forward path. The ROS need not prepare a NA response, since the destination of the response would be itself and testing the route optimization path again would be redundant.

The ROS marks route optimization target paths that pass these NUD tests as "reachable", and those that do not as "unreachable". These markings inform the AERO interface forwarding algorithm specified in Section 3.13.

Note that to avoid a DoS vector nodes MUST NOT return loopback NS(NUD) messages received from an unsecured link-layer source via a secured SPAN path.

3.19. Mobility Management and Quality of Service (QoS)

AERO is a Distributed Mobility Management (DMM) service. Each Server is responsible for only a subset of the Clients on the AERO link, as opposed to a Centralized Mobility Management (CMM) service where there is a single network mobility collective entity 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 default routing and mobility/multilink services for their dependent Clients. Clients are responsible for maintaining neighbor relationships with their Servers through periodic RS/RA exchanges, which also serves to confirm neighbor reachability. When a Client's underlying interface address and/or QoS information changes, the Client is responsible for updating the Server with this new information. Note that for Proxyed interfaces, however, the Proxy can also perform some RS/RA exchanges on the Client's behalf.

Mobility management considerations are specified in the following sections.

3.19.1. Mobility Update Messaging

Servers accommodate Client mobility/multilink and/or QoS change events by sending unsolicited NA (uNA) messages to each ROS in the target Client's Report List. When a Server sends a uNA message, it sets the IPv6 source address to the Client's AERO address, sets the destination address to All-Nodes multicast and sets the Target Address to the Client's Subnet-Router anycast address. The Server also includes an OMNI option with prefix registration information and with ifIndex-tuples for the target Client's remaining interfaces with S set to 0. The Server then includes a TLLAO with corresponding ifIndex-tuples with link layer addresses set to the corresponding target SPAN addresses. The Server sets the NA R flag to 1, the S flag to 0 and the O flag to 0, then encapsulates the message in a SPAN header with source set to its own SPAN address and destination set to the SPAN address of the ROS and sends the message into the SPAN.

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

When the ROS receives an uNA message, it ignores the message if there is no existing neighbor cache entry for the Client. Otherwise, it uses the included OMNI option and TLLAO information to update 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 uNA messages are lost, the ROS may be left with stale address and/or QoS information for the Client for up to REACHABLETIME seconds. During this time, the ROS can continue sending packets according to its stale neighbor cache information. When ReachableTime is close to expiring, the ROS will re-initiate route optimization and receive fresh link-layer address information.

In addition to sending uNA messages to the current set of ROSs for the Client, the Server also sends uNAs to the former link-layer address for any ifIndex-tuple for which the link-layer address has changed. The uNA messages update Proxys that cannot easily detect

(e.g., without active probing) when a formerly-active Client has departed.

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

When a Client needs to change its ANET addresses and/or QoS preferences (e.g., due to a mobility event), either the Client or its Proxys send RS messages to the Server via the SPAN with an OMNI option that includes an ifIndex-tuple with S set to 1 and with the new link quality and address information.

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

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

3.19.3. Bringing New Links Into Service

When a Client needs to bring new underlying interfaces into service (e.g., when it activates a new data link), it sends an RS message to the Server via the underlying interface with an OMNI option that includes an ifIndex-tuple with S set to 1 and appropriate link quality values and with link-layer address information for the new link.

3.19.4. Removing Existing Links from Service

When a Client needs to remove existing underlying interfaces from service (e.g., when it de-activates an existing data link), it sends an RS or uNA message to its Server with an OMNI option with appropriate link quality values.

If the Client needs to send RS/uNA messages over an underlying interface other than the one being removed from service, it MUST include ifIndex-tuples with appropriate link quality values for any underlying interfaces being removed from service.

3.19.5. Moving to a New Server

When a Client associates with a new Server, it performs the Client procedures specified in Section 3.15.2. The Client also includes MS-Release identifiers in the RS message OMNI option per [I-D.templin-6man-omni-interface] if it wants the new Server to notify any old Servers from which the Client is departing.

When the new Server receives the Client's RS message, it returns an RA as specified in Section 3.15.3 and sends up to MAX_NEIGHBOR_ADVERTIISEMENT uNA messages to any old Servers listed in OMNI option MS-Release identifiers. Each uNA message includes the Client's AERO address as the source address, the old Server's AERO address as the destination address, and an OMNI option with the Register/Release bit set to 0. The new Server wraps the uNA in a SPAN header with its own SPAN address as the source and the old Server's SPAN address as the destination, then sends the message into the SPAN.

When an old Server receives the uNA, it changes the Client's neighbor cache entry state to DEPARTED, sets the link-layer address of the Client to the new Server's SPAN address, and sets DepartTime to DEPARTTIME seconds. After a short delay (e.g., 2 seconds) the old Server withdraws the Client's MNP from the routing system. After DepartTime expires, the old Server deletes the Client's neighbor cache entry.

The old Server also sends unsolicited NA messages to all ROSs in the Client's Report List with an OMNI option with a single ifIndex-tuple with ifIndex set to 0 and S set to '1', and with the SPAN address of the new Server in a companion TLLAO. When the ROS receives the NA, it caches the address of the new Server in the existing asymmetric neighbor cache entry and marks the entry as STALE. Subsequent data packets will then flow according to any existing cached link-layer information and trigger a new NS(AR)/NA exchange via the new Server.

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

When a Client moves to a new Server, some of the fragments of a multiple fragment packet may have already arrived at the old Server while others are en route to the new Server, however no special attention in the reassembly algorithm is necessary when re-routed fragments are simply treated as loss.

3.20. Multicast

The AERO Client provides an IGMP (IPv4) [RFC2236] or MLD (IPv6) [RFC3810] proxy service for its EUNs and/or hosted applications [RFC4605]. The Client forwards IGMP/MLD messages over any of its underlying interfaces for which group membership is required. The IGMP/MLD messages may be further forwarded by a first-hop ANET access

router acting as an IGMP/MLD-snooping switch [RFC4541], then ultimately delivered to an AERO Proxy/Server acting as a Protocol Independent Multicast - Sparse-Mode (PIM-SM, or simply "PIM") Designated Router (DR) [RFC7761]. AERO Gateways also act as PIM routers (i.e., the same as AERO Proxys/Servers) on behalf of nodes on INET/EUN networks. The behaviors identified in the following sections correspond to Source-Specific Multicast (SSM) and Any-Source Multicast (ASM) operational modes.

3.20.1. Source-Specific Multicast (SSM)

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

For each S belonging to a prefix reachable via X's AERO interface, X originates a separate copy of the Join/Prune for each (S,G) in the message using its own AERO address as the source address and ALL-PIM-ROUTERS as the destination address. X then encapsulates each message in a SPAN header with source address set to the SPAN address of X and destination address set to S then forwards the message into the SPAN. The SPAN in turn forwards the message to AERO Server/Gateway "Y" that services S. At the same time, if the message was a Join, X sends a route-optimization NS message toward each S the same as discussed in Section 3.17. The resulting NAs will return the AERO address for the prefix that matches S as the network-layer source address and TLLAOs with the SPAN addresses corresponding to any ifIndex-tuples that are currently servicing S.

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

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

At some later time, Client C that holds an MNP for source S may depart from a first Proxy Z1 and/or connect via a new Proxy Z2. In that case, Y sends an unsolicited NA message to X the same as specified for unicast mobility in Section 3.19. When X receives the unsolicited NA message, it updates its asymmetric neighbor cache entry for the AERO address for source S and sends new Join messages to any new Proxys Z2. There is no requirement to send any Prune messages to old Proxys Z1 since source S will no longer source any multicast data traffic via Z1. Instead, the multicast state for (S,G) in Proxy Z1 will soon time out since no new Joins will arrive.

After some later time, C may move to a new Server Y2 and depart from old Sever Y1. In that case, Y1 sends Join messages for any of C's active (S,G) groups to Y2 while including its own AERO address as the source address. This causes Y2 to include Y1 in the multicast forwarding tree during the interim time that Y1's symmetric neighbor cache entry for C is in the DEPARTED state. At the same time, Y1 sends an unsolicited NA message to X with an OMNI option and TLLAO with ifIndex-tuple set to 0 and a release indication to cause X to release its asymmetric neighbor cache entry. X then sends a new Join message to S via the SPAN and re-initiates route optimization the same as if it were receiving a fresh Join message from a node on a downstream link.

3.20.2. Any-Source Multicast (ASM)

When an ROS X acting as a PIM router receives a Join/Prune from a node on its downstream interfaces containing one or more (*,G) pairs, it updates its Multicast Routing Information Base (MRIB) accordingly. X then forwards a copy of the message to the Rendezvous Point (RP) R for each G over the SPAN. X uses its own AERO address as the source address and ALL-PIM-ROUTERS as the destination address, then encapsulates each message in a SPAN header with source address set to the SPAN address of X and destination address set to R, then sends the message into the SPAN. At the same time, if the message was a Join X initiates NS/NA route optimization the same as for the SSM case discussed in Section 3.20.1.

For each source S that sends multicast traffic to group G via R, the Proxy/Server Z* for the Client that aggregates S encapsulates the packets in PIM Register messages and forwards them to R via the SPAN. R may then elect to send a PIM Join to Z* over the SPAN. This will result in an (S,G) tree rooted at Z* with R as the next hop so that R will begin to receive two copies of the packet; one native copy from the (S, G) tree and a second copy from the pre-existing (*, G) tree that still uses PIM Register encapsulation. R can then issue a PIM Register-stop message to suppress the Register-encapsulated stream. At some later time, if C moves to a new Proxy/Server Z*, it resumes sending packets via PIM Register encapsulation via the new Z*.

At the same time, as multicast listeners discover individual S's for a given G, they can initiate an (S,G) Join for each S under the same procedures discussed in Section 3.20.1. Once the (S,G) tree is established, the listeners can send (S, G) Prune messages to R so that multicast packets for group G sourced by S will only be delivered via the (S, G) tree and not from the (*, G) tree rooted at R. All mobility considerations discussed for SSM apply.

3.20.3. Bi-Directional PIM (BIDIR-PIM)

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

3.21. Operation over Multiple AERO Links (VLANs)

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

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

Each AERO link could utilize the same or different ANET connections. The links can be distinguished at the link-layer via Virtual Local Area Network (VLAN) tagging (e.g., IEEE 802.1Q) and/or through assignment of distinct sets of MSPs on each link. This gives rise to the opportunity for supporting multiple redundant networked paths,

where each VLAN is distinguished by a different label (e.g., colors such as Red, Green, Blue, etc.). In particular, the Client can tag its RS messages with the appropriate label to cause the network to select the desired VLAN.

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

In a first alternative, if each AERO link services different MSPs, then the Client can receive a distinct MNP from each of the links. IP routing will therefore assure that the correct Red/Green/Blue/etc. network is used for both outbound and inbound traffic. This can be accomplished using existing technologies and approaches, and without requiring any special supporting code in correspondent nodes or Relays.

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

3.22. DNS Considerations

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

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

3.23. Transition Considerations

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

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

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

3.24. Detecting and Reacting to Server and Relay Failures

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

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

Proxys SHOULD use proactive NUD for Servers for which there are currently active ANET Clients in a manner that parallels BFD, i.e., by sending unicast NS messages in rapid succession to receive solicited NA messages. When the Proxy is also sending RS messages on behalf of ANET Clients, the RS/RA messaging can be considered as

equivalent hints of forward progress. This means that the Proxy need not also send a periodic NS if it has already sent an RS within the same period. If a Server fails, the Proxy will cease to receive advertisements and can quickly inform Clients of the outage by sending multicast RA messages on the ANET interface.

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

4. Implementation Status

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

5. IANA Considerations

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

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

No further IANA actions are required.

6. Security Considerations

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

AERO Servers, Gateways and Proxys targeted by a route optimization may also receive packets directly from the INET partitions instead of via the SPAN. For INET partitions that apply effective ingress filtering to defeat source address spoofing, the simple data origin authentication procedures in Section 3.11 can be applied.

For INET partitions that cannot apply effective ingress filtering, the two options for securing communications include 1) disable route optimization so that all traffic is conveyed over secured tunnels via the SPAN, or 2) enable on-demand secure tunnel creation between INET partition neighbors. Option 1) would result in longer routes than necessary and traffic concentration on critical infrastructure elements. Option 2) could be coordinated by establishing a secured tunnel on-demand instead of performing an NS/NA exchange in the route optimization procedures. Procedures for establishing on-demand secured tunnels are out of scope.

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

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

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

AERO Gateways must implement ingress filtering to avoid a spoofing attack in which spurious SPAN messages are injected into an AERO link from an outside attacker. AERO Clients MUST ensure that their connectivity is not used by unauthorized nodes on their EUNs to gain

access to a protected network, i.e., AERO Clients that act as routers MUST NOT provide routing services for unauthorized nodes. (This concern is no different than for ordinary hosts that receive an IP address delegation but then "share" the address with other nodes via some form of Internet connection sharing such as tethering.)

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

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

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

7. Acknowledgements

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

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

Benson, Katie Tran and Eric Yeh are especially acknowledged for implementing the AERO functions as extensions to the public domain OpenVPN distribution.

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

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

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

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

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

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

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

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

8. References

8.1. Normative References

[I-D.templin-6man-omni-interface]
Templin, F. and T. Whyman, "Transmission of IPv6 Packets over Overlay Multilink Network (OMNI) Interfaces", draft-templin-6man-omni-interface-07 (work in progress), March 2020.

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

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

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

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

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

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

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

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

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

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

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

8.2. Informative References

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

[I-D.ietf-6man-segment-routing-header] Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-26 (work in progress), October 2019.

[I-D.ietf-dmm-distributed-mobility-anchoring] Chan, A., Wei, X., Lee, J., Jeon, S., and C. Bernardos, "Distributed Mobility Anchoring", draft-ietf-dmm-distributed-mobility-anchoring-15 (work in progress), March 2020.

[I-D.ietf-intarea-gue] Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-intarea-gue-09 (work in progress), October 2019.

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

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

[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-05 (work in progress), January 2020.

[I-D.templin-6man-dhcpv6-ndopt]
Templin, F., "A Unified Stateful/Stateless Configuration Service for IPv6", draft-templin-6man-dhcpv6-ndopt-09 (work in progress), January 2020.

[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-25 (work in progress), January 2020.

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

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

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

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

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

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

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

[RFC2492] Armitage, G., Schulter, P., and M. Jork, "IPv6 over ATM Networks", RFC 2492, DOI 10.17487/RFC2492, January 1999, <https://www.rfc-editor.org/info/rfc2492>.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <https://www.rfc-editor.org/info/rfc4541>.

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

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

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

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

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

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

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

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

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

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

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

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

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

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

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

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

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

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <https://www.rfc-editor.org/info/rfc6438>.

[RFC6706] Templin, F., Ed., "Asymmetric Extended Route Optimization (AERO)", RFC 6706, DOI 10.17487/RFC6706, August 2012, <https://www.rfc-editor.org/info/rfc6706>.

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

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

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

[RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <https://www.rfc-editor.org/info/rfc7421>.

[RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <https://www.rfc-editor.org/info/rfc7761>.

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

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

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

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

Appendix A. AERO Alternate Encapsulations

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

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

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

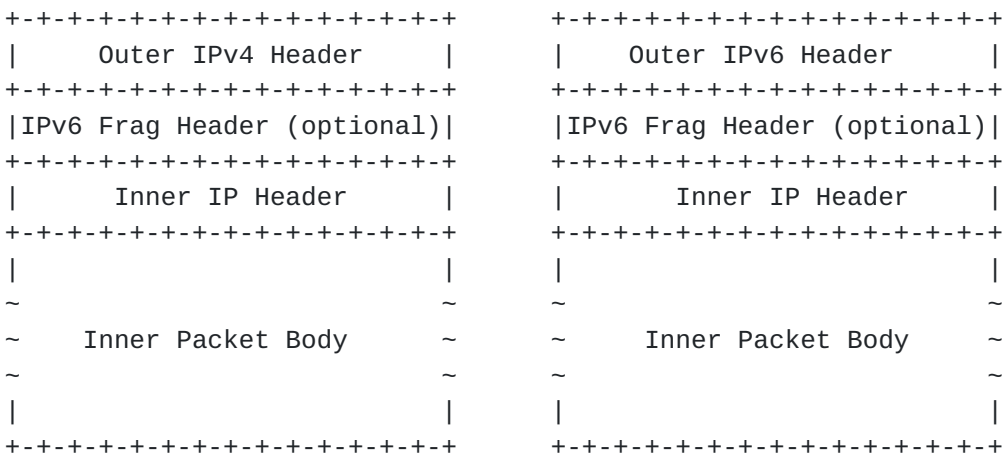

Figure 6: Minimal Encapsulation Format using IP-in-IP

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

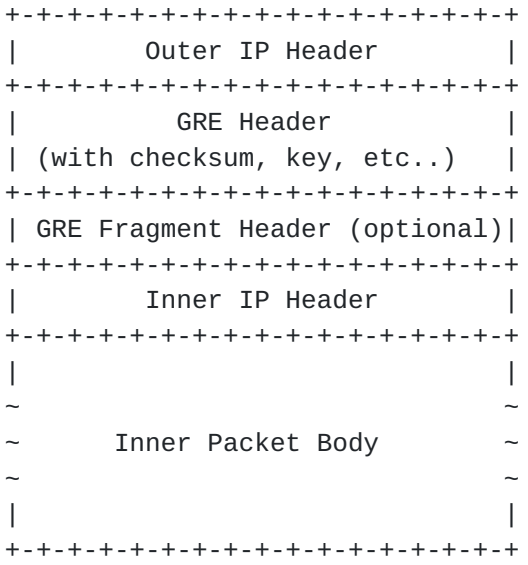

Figure 7: Minimal Encapsulation Using GRE

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

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

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

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

Appendix B. Non-Normative Considerations

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

B.1. Implementation Strategies for Route Optimization

Route optimization as discussed in Section 3.17 results in the route optimization source (ROS) creating an asymmetric neighbor cache entry for the target neighbor. The neighbor cache entry is maintained for at most REACHABLETIME seconds and then deleted unless updated. In order to refresh the neighbor cache entry lifetime before the ReachableTime timer expires, the specification requires implementations to issue a new NS/NA exchange to reset ReachableTime to REACHABLETIME seconds while data packets are still flowing. However, the decision of when to initiate a new NS/NA exchange and to perpetuate the process is left as an implementation detail.

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

The monitoring of the neighbor data packet traffic therefore becomes an asymmetric ongoing process during the neighbor cache entry lifetime. If the neighbor cache entry expires, future data packets

will trigger a new NS/NA exchange while the packets themselves are delivered over a longer path until route optimization state is re-established.

B.2. Implicit Mobility Management

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

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

B.3. Direct Underlying Interfaces

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

B.4. AERO Clients on the Open Internetwork

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

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

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

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

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

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

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

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

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

of the value in the Subnet Prefix field) to match against the IPv6 source address of the ND message.

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

B.7. AERO Critical Infrastructure Considerations

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

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

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

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

B.8. AERO Server Failure Implications

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

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

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

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

B.9. AERO Client / Server Architecture

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

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

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

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

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

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

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

In contrast to the Google and IETF philosophies, Yahoo and Amazon both provide clients with a (short) list of IP addresses with Yahoo

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

The same implications observed for common highly-available services in the Internet apply also to the AERO client/server architecture. When an AERO Client connects to one or more ANETs, it discovers one or more AERO Server addresses through the mechanisms discussed in earlier sections. Each Server address presumably leads to a fault-tolerant clustering arrangement such as supported by Linux-HA, Extended Virtual Synchrony or Paxos. Such an arrangement has precedence in common Internet service deployments in lightweight virtual machines without requiring expensive hardware deployment. Similarly, common Internet service deployments set service IP addresses on service distribution points that may relay requests to many different servers.

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

Appendix C. Change Log

<< RFC Editor - remove prior to publication >>

Changes from draft-templin-intarea-6706bis-32 to draft-templin-intrea-6706bis-33:

- Updated Proxy discussion with "point-to-multipoint" server coordination

- Significant updates to Address Resolution and NUD to include correct addresses in messages

- Differentiate between NS(AR) and NS(NUD) as their addresses and use cases differ.

Changes from draft-templin-intarea-6706bis-30 to draft-templin-intrea-6706bis-31:

- Added "advisory PTB messages" under FAA SE2025 contract number DTFAWA-15-D-00030.

Changes from draft-templin-intarea-6706bis-29 to draft-templin-intrea-6706bis-30:

- Deprecate "primary" concept. Now, RS/RA keepalives are mainatained over *all* underlying interfaces (i.e., and not just one primary).

Changes from draft-templin-intarea-6706bis-28 to draft-templin-intrea-6706bis-29:

- Changed OMNI interface citation to "draft-templin-6man-omni-interface"

- Changed SPAN Service Prefix to fd80::/10.

- Changed S/TLLAO format to include 'S' bit for ifIndex corresponding to the underlying interface that is Source of ND message.

- Updated Path MTU

Changes from draft-templin-intarea-6706bis-27 to draft-templin-intrea-6706bis-28:

- MTU and fragmentation.

Changes from draft-templin-intarea-6706bis-26 to draft-templin-intrea-6706bis-27:

- MTU and fragmentation.

- SPAN Service Prefix set to fd00::/10

- Client SPAN addresses defined.

Changes from draft-templin-intarea-6706bis-25 to draft-templin-intrea-6706bis-26:

- MTU and RA configuration information updated.

Changes from draft-templin-intarea-6706bis-24 to draft-templin-intrea-6706bis-25:

- Added concept of "primary" to allow for proxyed RS/RA over only selected underlying interfaces.

- General Cleanup.

Changes from draft-templin-intarea-6706bis-23 to draft-templin-intrea-6706bis-24:

- OMNI interface spec now a normative reference.

- Use REACHABLETIME as the nominal Router Lifetime to return in RAs.

- General cleanup.

Changes from draft-templin-intarea-6706bis-22 to draft-templin-intrea-6706bis-23:

- Choice of using either RS/RA or unsolicited NA for old Server notification.

- General cleanup.

Changes from draft-templin-intarea-6706bis-21 to draft-templin-intrea-6706bis-22:

- Tightened up text on Proxy.

- Removed unnecessarily restrictive texts.

- General cleanup.

Changes from draft-templin-intarea-6706bis-20 to draft-templin-intrea-6706bis-21:

- Clarified relationship between OMNI and S/TLLAO ifIndex-tuples.

- Important text in Section 13.15.3 on Servers timing out Clients that have gone silent without sending a departure notification.

- New text on RS/RA as "hints of forward progress" for proactive NUD.

Changes from draft-templin-intarea-6706bis-19 to draft-templin-intrea-6706bis-20:

- Included new route optimization source and destination addressing strategy. Now, route optimization maintenance uses the address of the existing Server instead of the data packet destination address so that less pressure is placed on the BGP routing system convergence time and Server constancy is supported.

- Included new method for releasing from old MSE without requiring Client messaging.

- Included references to new OMNI interface spec (including the OMNI option).

- New appendix on AERO Client/Server architecture.

Changes from draft-templin-intarea-6706bis-18 to draft-templin-intrea-6706bis-19:

- Changed Proxy/Server keepalives to use "proactive NUD" in a manner tha paralles BFD

Changes from draft-templin-intarea-6706bis-17 to draft-templin-intrea-6706bis-18:

- Discuss how AERO option is used in relation to S/TLLAOs

- New text on Bidirectional Forwarding Detection (BFD)

- Cleaned up usage (and non-usage) of unsolicited NAs

- New appendix on Server failures

Changes from draft-templin-intarea-6706bis-15 to draft-templin-intrea-6706bis-17:

- S/TLLAO now includes multiple link-layer addresses within a single option instead of requiring multiple options

- New unsolicited NA message to inform the old link that a Client has moved to a new link

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

- MTU and fragmentation

- New details in movement to new Server

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

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

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

- New paragraph in Section 3.6 on AERO interface layering over secured tunnels
- Removed extraneous text in Section 3.7
- Added new detail to the forwarding algorithm in Section 3.9
- Clarified use of fragmentation
- Route optimization now supported for both MNP and non-MNP-based prefixes
- Relays are now seen as link-layer elements in the architecture.
- Built out multicast section in detail.
- New Appendix on implementation considerations for route optimization.

Changes from draft-templin-intarea-6706bis-11 to draft-templin-intrea-6706bis-12:

- Introduced Gateways as a new AERO element for connecting Correspondent Nodes on INET links
- Introduced terms "Access Network (ANET)" and "Internetwork (INET)"
- Changed "ASP" to "MSP", and "ACP" to "MNP"
- New figure on the relation of Segments to the SPAN and AERO link
- New "S" bit in S/TLLAO to indicate the "Source" S/TLLAO as opposed to additional S/TLLAOs
- Changed Interface ID for Servers from 255 to 0xffff
- Significant updates to Route Optimization, NUD, and Mobility Management

- New Section on Multicast
- New Section on AERO Clients in the open Internetwork
- New Section on Operation over multiple AERO links (VLANs over the SPAN)
- New Sections on DNS considerations and Transition considerations
-

Changes from draft-templin-intarea-6706bis-10 to draft-templin-intrea-6706bis-11:

- Added The SPAN

Changes from draft-templin-intarea-6706bis-09 to draft-templin-intrea-6706bis-10:

- Orphaned packets in flight (e.g., when a neighbor cache entry is in the DEPARTED state) are now forwarded at the link layer instead of at the network layer. Forwarding at the network layer can result in routing loops and/or excessive delays of forwarded packets while the routing system is still reconverging.
- Update route optimization to clarify the unsecured nature of the first NS used for route discovery
- Many cleanups and clarifications on ND messaging parameters

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

- Changed PRL to "MAP list"
- For neighbor cache entries, changed "static" to "symmetric", and "dynamic" to "asymmetric"
- Specified Proxy RS/RA exchanges with Servers on behalf of Clients
- Added discussion of unsolicited NAs in Section 3.16, and included forward reference to Section 3.18
- Added discussion of AERO Clients used as critical infrastructure elements to connect fixed networks.
- Added network-based VPN under security considerations

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

- New section on AERO-Aware Access Router

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

- Added "R" bit for release of PDs. Now have a full RS/RA service that can do PD without requiring DHCPv6 messaging over-the-air
- Clarifications on solicited vs unsolicited NAs
- Clarified use of MAX_NEIGHBOR_ADVERTISEMENTS for the purpose of increase reliability

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

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

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

- Note on encapsulation avoidance in Section 4.

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

- Remove DHCPv6 Server Release procedures that leveraged the old way Relays used to "route" between Server link-local addresses
- Remove all text relating to Relays needing to do any AERO-specific operations
- 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.
- 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.
- Emphasize distributed mobility management
- AERO address-based RS injection of ACP into underlying routing system.

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

- 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.

- 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.

- 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