Network Working Group
Internet-Draft
Obsoletes: rfc5320, rfc5558, rfc5720, rfc6179, rfc6706 (if approved)
Intended status: Standards Track
Expires: December 31, 2020

F. Templin, Ed.
Boeing Research & Technology

June 29, 2020

Asymmetric Extended Route Optimization (AERO)
draft-templin-intarea-6706bis-58

Abstract

This document specifies the operation of IP over Overlay Multilink Network (OMNI) interfaces using the Asymmetric Extended Route Optimization (AERO) internetworking and mobility management service. AERO uses 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, intelligent transportation systems, mobile Virtual Private Networks (VPNs) and many other applications.

Table of Contents

1. Introduction 4
2. Terminology 5
3. Asymmetric Extended Route Optimization (AERO) 10
3.1. AERO Node Types 10
3.2. The AERO Service over OMNI Links 11
3.2.1. AERO/OMNI Reference Model 11
3.2.2. Link-Local Addresses (LLAs) and Unique Local Addresses (ULAs) 14
3.2.3. AERO Routing System 15
3.2.4. AERO Encapsulation 16
3.2.5. Segment Routing Topologies (SRTs) 18
3.2.6. Segment Routing To the OMNI Link 18
3.2.7. Segment Routing Within the OMNI Link 19
3.2.8. Segment Routing Header Compression 21
3.3. OMNI Interface Characteristics 21
3.4. OMNI Interface Initialization 25
3.4.1. AERO Server/Relay Behavior 25
3.4.2. AERO Proxy Behavior 26
3.4.3. AERO Client Behavior 26
3.4.4. AERO Bridge Behavior 26
3.5. OMNI Interface Neighbor Cache Maintenance 26
3.6. OMNI Interface Encapsulation and Re-encapsulation 28
3.7. OMNI Interface Decapsulation 30
3.8. OMNI Interface Data Origin Authentication 30
3.9. OMNI Interface MTU and Fragmentation 30
3.10. OMNI Interface Forwarding Algorithm 30
3.10.1. Client Forwarding Algorithm 31
3.10.2. Proxy Forwarding Algorithm 32
3.10.3. Server/Relay Forwarding Algorithm 33
3.10.4. Bridge Forwarding Algorithm 34
3.11. OMNI Interface Error Handling 35

3.12. AERO Router Discovery, Prefix Delegation and Autoconfiguration 37
3.12.1. AERO ND/PD Service Model 37
3.12.2. AERO Client Behavior 38
3.12.3. AERO Server Behavior 40
3.13. The AERO Proxy 43
3.13.1. Ancillary Servers Acting as Proxies 45
3.13.2. Detecting and Responding to Server Failures 45
3.13.3. Point-to-Multipoint Server Coordination 46
3.14. AERO Route Optimization / Address Resolution 47
3.14.1. Route Optimization Initiation 47
3.14.2. Relaying the NS 48
3.14.3. Processing the NS and Sending the NA 48
3.14.4. Relaying the NA 49
3.14.5. Processing the NA 49
3.14.6. Route Optimization Maintenance 49
3.15. Neighbor Unreachability Detection (NUD) 50
3.16. Mobility Management and Quality of Service (QoS) 52
3.16.1. Mobility Update Messaging 52
3.16.2. Announcing Link-Layer Address and/or QoS Preference Changes 53
3.16.3. Bringing New Links Into Service 54
3.16.4. Removing Existing Links from Service 54
3.16.5. Moving to a New Server 54
3.17. Multicast 55
3.17.1. Source-Specific Multicast (SSM) 55
3.17.2. Any-Source Multicast (ASM) 57
3.17.3. Bi-Directional PIM (BIDIR-PIM) 58
3.18. Operation over Multiple OMNI Links 58
3.19. DNS Considerations 59
3.20. Transition Considerations 59
3.21. Detecting and Reacting to Server and Bridge Failures . . 60
3.22. AERO Clients on the Open Internet 60
3.22.1. Use of SEND and CGA 63
3.23. Time-Varying MNPs 64
4. Implementation Status 65
5. IANA Considerations 65
6. Security Considerations 65
7. Acknowledgements 67
8. References 69
8.1. Normative References 69
8.2. Informative References 70
Appendix A. Non-Normative Considerations 76
A.1. Implementation Strategies for Route Optimization 76
A.2. Implicit Mobility Management 76
A.3. Direct Underlying Interfaces 77
A.4. AERO Critical Infrastructure Considerations 77
A.5. AERO Server Failure Implications 78

A.6. AERO Client / Server Architecture 78
Appendix B. Change Log 80
Author's Address 81

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 such as intelligent transportation systems. AERO is an internetworking and mobility management service based on the Overlay Multilink Network Interface (OMNI) [I-D.templin-6man-omni-interface] Non-Broadcast, Multiple Access (NBMA) virtual link model. The OMNI link is a virtual overlay configured over one or more underlying Internetworks, and nodes on the link can exchange IP packets via tunneling. Multilink operation allows for increased reliability, bandwidth optimization and traffic path diversity.

The AERO service comprises Clients, Proxys, Servers and Relays that are seen as OMNI link neighbors as well as Bridges that interconnect OMNI link segments. Each node's OMNI interface uses an IPv6 link-local address format that supports operation of the IPv6 Neighbor Discovery (ND) protocol [RFC4861] and links ND to IP forwarding. A node's OMNI interface can be configured over multiple underlying interfaces, and 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 provides a cloud-based service where mobile nodes may use any Server acting as a Mobility Anchor Point (MAP) and fixed nodes may use any Relay on the link for efficient communications. Fixed nodes forward packets destined to other AERO nodes to the nearest Relay, 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 Bridges are interconnected in a secured private BGP overlay routing instance using encapsulation to provide a hybrid routing/bridging service that joins the underlying Internetworks of multiple disjoint administrative domains into a single unified OMNI link. Each OMNI link instance is characterized by the set of Mobility

Service Prefixes (MSPs) common to all mobile nodes. The link extends to the point where a Relay/Server is on the optimal route from any correspondent node on the link, and provides a conduit between the underlying Internetwork and the OMNI link. To the underlying Internetwork, the Relay/Server is the source of a route to the MSP, and hence uplink traffic to the mobile node is naturally routed to the nearest Relay/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]. It can also be used to facilitate vehicular and pedestrian communications services for intelligent transportation systems. Other applicable use cases are also in scope.

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

2. Terminology

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

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

IPv6 Prefix Delegation (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. 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, etc.) that often provides link-layer security services such as IEEE 802.1X and physical-layer security prevent unauthorized access internally and with border network-layer security services such as firewalls and proxies that prevent unauthorized outside access.

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

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

INET 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. (Each INET partition is seen as a separate OMNI link segment as 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.

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

OMNI link
: the same as defined in [I-D.templin-6man-omni-interface], and manifested by IPv6 encapsulation [RFC2473]. The OMNI link spans underlying INET segments joined by virtual bridges in a spanning

tree the same as a bridged campus LAN. AERO nodes on the OMNI link appear as single-hop neighbors even though they may be separated by multiple underlying INET hops, and can use Segment Routing [RFC8402] to cause packets to visit selected waypoints on the link.

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

OMNI Link-Local Address (LLA)
a link local IPv6 address per [RFC4291] constructed as specified in Section 3.2.2.

OMNI Unique-Local Address (ULA)
a unique local IPv6 address per [RFC4193] constructed as specified in Section 3.2.2. OMNI ULAs are statelessly derived from OMNI LLAs, and vice-versa.

underlying interface
an ANET or INET interface over which an OMNI interface is configured.

Mobility Service Prefix (MSP)
an IP prefix assigned to the OMNI 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 Relay.

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

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

AERO Server ("Server")
an INET node that configures an OMNI interface to provide default forwarding and mobility/multilink services for AERO Clients. The

Server assigns an administratively-provisioned LLA to its OMNI interface to support the operation of the ND/PD services, and advertises all of its associated MNPs via BGP peerings with Bridges.

AERO Relay ("Relay")
: an AERO Server that also provides forwarding services between nodes reached via the OMNI link and correspondents on other links. AERO Relays 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 Relay advertises the MSP(s) to its downstream networks, and distributes all of its associated MNPs and non-MNP IP routes via BGP peerings with Bridges (i.e., the same as for an AERO Server).

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

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].

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

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

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

network layer address

the source or destination address of an encapsulated IP packet presented to the OMNI interface.

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

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

Mobile Router (MR)
: a MN's on-board router that forwards packets between any downstream-attached networks and the OMNI 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 Relay for a non-MNP destination.

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

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

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

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

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

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

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

3. Asymmetric Extended Route Optimization (AERO)

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

3.1. AERO Node Types

AERO Bridges provide hybrid routing/bridging services (as well as a security trust anchor) for nodes on an OMNI link. Bridges use standard IPv6 routing to forward packets both within the same INET partitions and between disjoint INET partitions based on a mid-layer IPv6 encapsulation per [RFC2473]. The inner IP layer experiences a virtual bridging service since the inner IP TTL/Hop Limit is not decremented during forwarding. Each Bridge also peers with Servers and other Bridges in a dynamic routing protocol instance to provide a Distributed Mobility Management (DMM) service for the list of active MNPs (see Section 3.2.3). Bridges present the OMNI link as a set of one or more Mobility Service Prefixes (MSPs) and configure secured tunnels with Servers, Relays, Proxys and other Bridges; 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 Bridges in a dynamic routing protocol instance to advertise its list of associated MNPs (see Section 3.2.3). Servers facilitate PD exchanges with Clients, where each delegated prefix becomes an MNP

taken from an MSP. Servers forward packets between OMNI interface neighbors and track each Client's mobility profiles.

AERO Clients register their MNPs through PD exchanges with AERO Servers over the OMNI 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 Clients to associate with 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 OMNI link according to forwarding information in the neighbor cache. The Proxy function is specified in Section 3.13.

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

AERO Bridges, Servers, Proxys and Relays 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 underlying interfaces with addresses that may change when the Client moves to a new network connection point.

3.2. The AERO Service over OMNI Links

3.2.1. AERO/OMNI Reference Model

Figure 1 presents the basic OMNI link reference model:

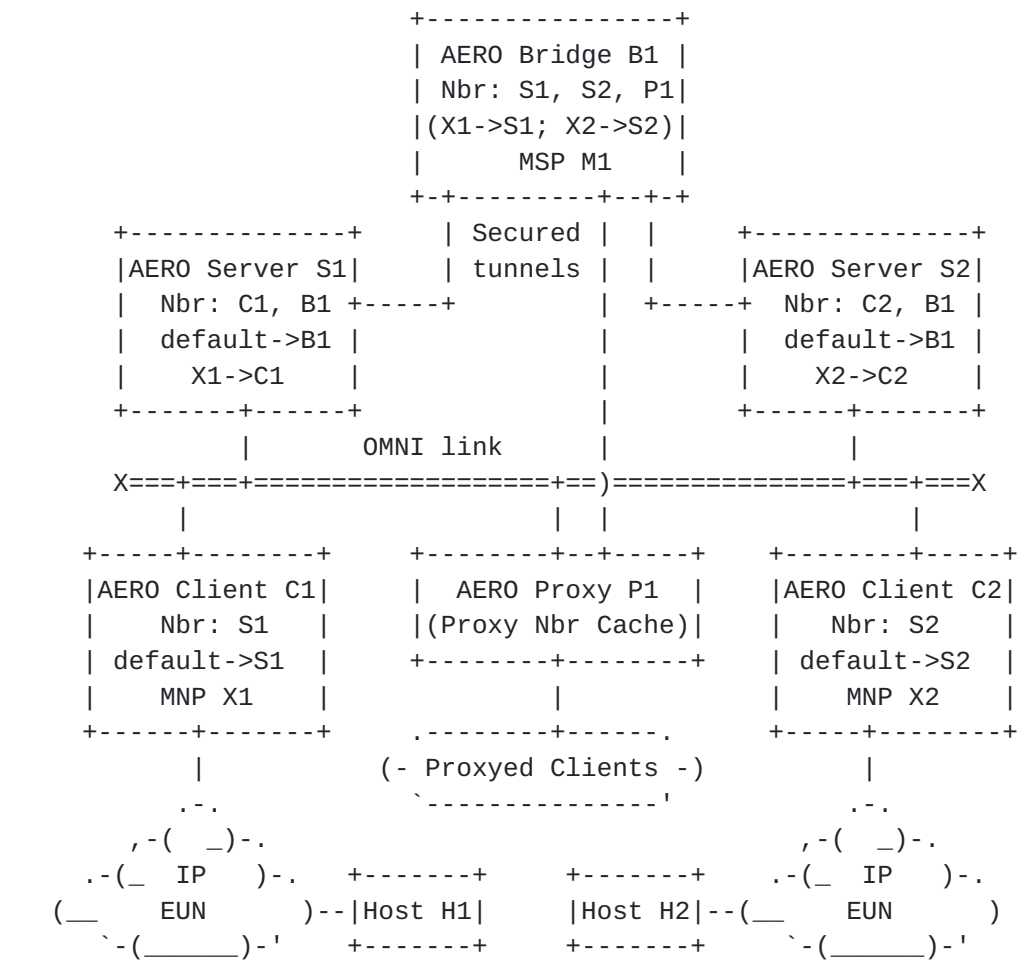

Figure 1: AERO/OMNI Reference Model

In this model:

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

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

- 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 Bridge B1 and provides proxy services for AERO Clients in secured enclaves that cannot associate directly with other OMNI link neighbors.

An OMNI 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 OMNI 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 Bridges.

The same as for traditional campus LANs, multiple OMNI link segments can be joined into a single unified link via a virtual bridging service using a mid-layer IPv6 encpasulation per [RFC2473] known as the "SPAN header" that supports inter-segment forwarding (i.e., bridging) without decrementing the network-layer TTL/Hop Limit. This bridging of OMNI link segments is shown in Figure 2:

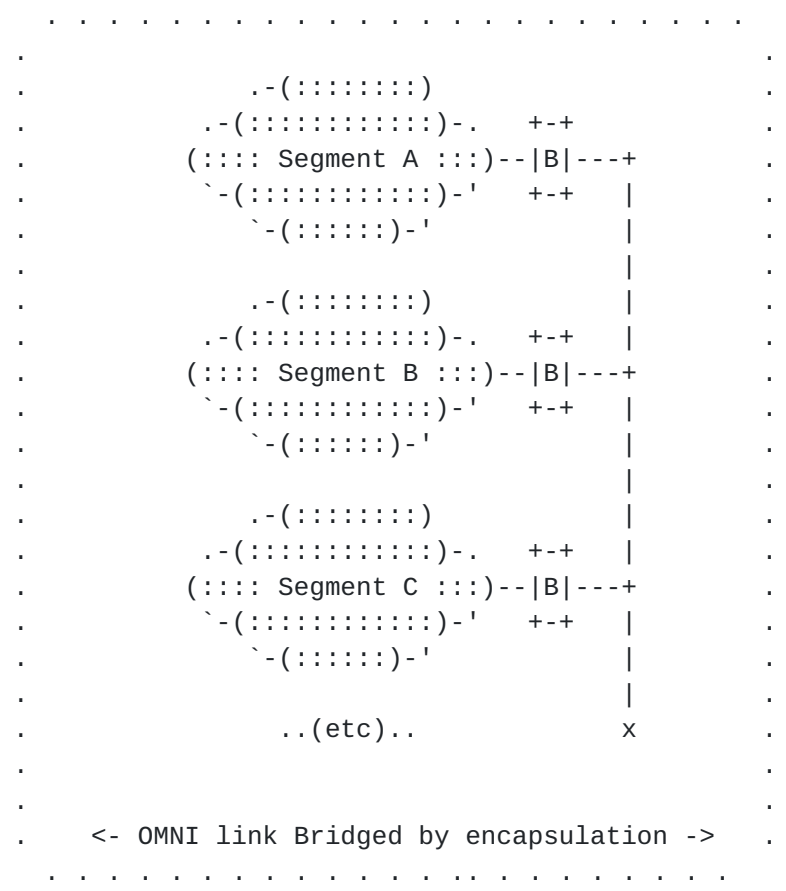

Figure 2: Bridging OMNI Link Segments

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

3.2.2. Link-Local Addresses (LLAs) and Unique Local Addresses (ULAs)

AERO nodes on OMNI links use the Link-Local Address (LLA) prefix fe80::/10 [RFC4193] to assign LLAs used for network-layer addresses in IPv6 ND and data messages. They also use the Unique Local Address (ULA) prefix fc80::/10 [RFC4193] to form ULAs used for SPAN header source and desitnation addresses. See [I-D.templin-6man-omni-interface] for a full specification of the LLAs and ULAs used by AERO nodes on OMNI links.

For routing system organization (see Section 3.2.3), ULAs are organized in partition prefixes, e.g., fc80::1000/116. For each such partition prefix, the Bridge(s) that connect that segment assign the all-zero's address of the prefix as a Subnet Router Anycast address. For example, the Subnet Router Anycast address for fc80::1000/116 is simply fc80::1000.

3.2.3. AERO Routing System

The AERO routing system comprises a private instance of the Border Gateway Protocol (BGP) [RFC4271] that is coordinated between Bridges and 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 Bridges but does not peer with other Servers. Each INET of a multi-segment OMNI link must include one or more Bridges, which peer with the Servers and Proxys within that INET. All Bridges 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 Bridges of different INETs peer with one another using eBGP.

Bridges advertise the OMNI 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 Bridge configures a black-hole route for each of its MSPs. By black-holing the MSPs, the Bridge 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 Bridges which have full topology knowledge.

Each OMNI link segment assigns a unique sub-prefix of fc80::/96 known as the ULA partition prefix. For example, a first segment could assign fc80::1000/116, a second could assign fc80::2000/116, a third could assign fc80::3000/116, etc. The administrative authorities for each segment must therefore coordinate to assure mutually-exclusive partiton prefix assignments, but internal provisioning of each prefix is an independent local consideration for each administrative authority.

ULA partition prefixes are statitcally represented in Bridge forwarding tables. Bridges join multiple segments into a unified

OMNI link over multiple diverse administrative domains. They support a bridging function by first establishing forwarding table entries for their partiion prefixes either via standard BGP routing or static routes. For example, if three Bridges ('A', 'B' and 'C') from different segments serviced fc80::1000/116, fc80::2000/116 and fc80::3000/116 respectively, then the forwarding tables in each Bridge are as follows:

A: fc80::1000/116->local, fc80::2000/116->B, fc80::3000/116->C

B: fc80::1000/116->A, fc80::2000/116->local, fc80::3000/116->C

C: fc80::1000/116->A, fc80::2000/116->B, fc80::3000/116->local

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

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

D: fc80:2001:db8:1000:2000::/72

E: fc80:2001:db8:3000:4000::/72

F: fc80:2001:db8:5000:6000::/72

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

3.2.4. AERO Encapsulation

With the Client and partition prefixes in place in each Bridge's forwarding table, control and data packets sent between AERO nodes in different segments can therefore be carried over the via mid-layer encapsulation using the SPAN header. For example, when a source AERO node forwards a packet with IPv6 address 2001:db8:1:2::1 to a target AERO node with IPv6 address 2001:db8:1000:2000::1, it first encapsulates the packet in a SPAN header with source address set to fc80:2001:db8:1:2:: and destination address set to fc80:2001:db8:1000:2000::. Next, it encapsulates the resulting SPAN packet 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 Bridge (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:

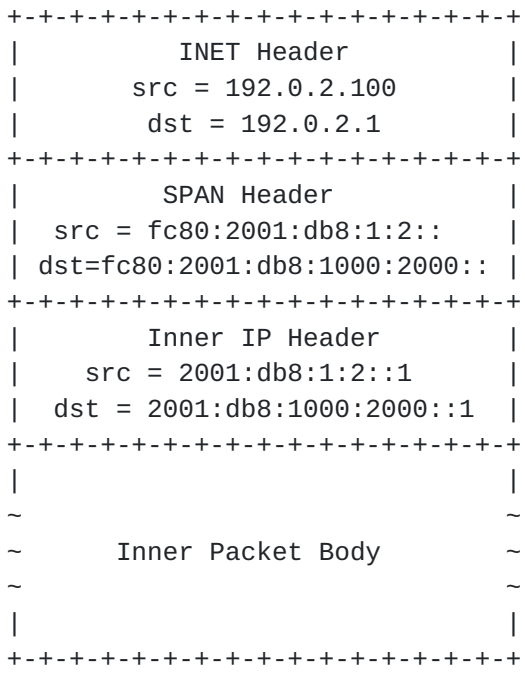

Figure 3: SPAN Encapsulation

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 as discussed in Section 3.6.

This gives rise to a routing system that contains both Client prefix routes that may change dynamically due to regional node mobility and partion prefix routes that never change. The Bridges can therefore provide link-layer bridging by sending packets over the spanning tree 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.

In normal operations, IPv6 ND messages are conveyed over secured paths between OMNI link neighbors so that specific Proxys, Servers or Relays can be addressed without being subject to mobility events. Conversely, only the first few packets destined to Clients need to traverse secured paths until route optimization can determine a more direct path.

3.2.5. Segment Routing Topologies (SRTs)

The 16-bit sub-prefixes of fc80::/10 identify up to 64 distinct Segment Routing Topologies (SRTs). Each SRT is a mutually-exclusive OMNI link overlay instance using a mutually-exclusive set of ULAs, and emulates a Virtual LAN (VLAN) service for the OMNI link. In some cases (e.g., when redundant topologies are needed for fault tolerance and reliability) it may be beneficial to deploy multiple SRTs that act as independent overlay instances. A communication failure in one instance therefore will not affect communications in other instances.

Each SRT is identified by a distinct value in bits 10-15 of fc80::10, i.e., as fc80::/16, fc81::/16, fc82::/16, etc. This document asserts that up to four SRTs provide a level of safety sufficient for critical communications such as civil aviation. Each SRT is designated with a color that identifies a different OMNI link instance as follows:

- Red - corresponds to fc80::/16
- Green - corresponds to fc81::/16
- Blue-1 - corresponds to fc82::/16
- Blue-2 - corresponds to fc83::/16
- fc84::/16 through fcbf::/16 are available for additional SRTs.

Each OMNI interface assigns an anycast ULA corresponding to its SRT prefix. For example, the anycast ULA for the Green SRT is simply fc81::. The anycast ULA is used for OMNI interface determination in Safety-Based Multilink (SBM) as discussed in [I-D.templin-6man-omni-interface]. Each OMNI interface further applies Performance-Based Multilink (PBM) internally.

3.2.6. Segment Routing To the OMNI Link

An original IPv6 source can direct a packet to an OMNI link Client by including a Segment Routing Header (SRH) with the anycast ULA for the selected SRT as either the IPv6 destination or as an intermediate hop within the SRH. This allows the original source to determine the specific topology a packet will traverse when there may be multiple alternatives to choose from. Since the SRH contains no useful information for the destination, the Client may elect to delete the SRH before forwarding in order to reduce overhead. This form of Segment Routing supports Safety-Based Multilink (SBM), and can be exercised through general-purpose SRH types such as [RFC8754].

3.2.7. Segment Routing Within the OMNI Link

AERO nodes that insert a SPAN header can use Segment Routing within the OMNI link when necessary to influence the path of packets destined to targets in remote segments without requiring all packets to traverse strict spanning tree paths.

When a Client, Proxy or Server has a packet to send to a target discovered through route optimization located in the same OMNI link segment, it encapsulates the packet in a SPAN header with the ULA of the target as the destination address if fragmentation is necessary; otherwise, it may omit the SPAN header. The node then uses the target's Link Layer Address (L2ADDR) information for INET encapsulation without including an SRH.

When a Client, Proxy or Server has a packet to send to a route optimization target located in a remote OMNI link segment, it encapsulates the packet in a SPAN header with its own ULA as the source address. The node then SHOULD include an SRH [RFC8754] while forwarding the packet to a Bridge.

When the SRH is omitted, the node sets the destination address to the ULA of the target Client/Proxy/Server and packet forwarding is via spanning tree paths. When the SRH is included, the node first sets the destination address to the ULA Subnet Router Anycast address of the remote segment and sets the ULA of the target's Proxy/Server as the Last Hop Segment (LHS) ID. The node also includes an AERO Route Optimization specification in the SRH TLV section as shown in Figure 4:

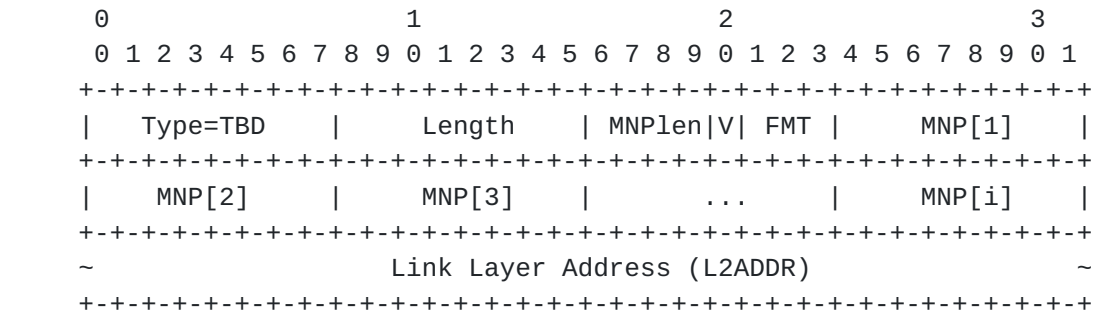

Figure 4: AERO Route Optimization SRH TLV

In this format:

- Type is TBD to be assigned according to the Segment Routing Header TLV registry [RFC8754].

- Length is the length of the body of the TLV in bytes, excluding the Type and Length fields.

- MNPlen encodes a value 'i' (between 0 and 15) that indicates the number of octets of the IPv4/IPv6 MNP prefix that follows.

- V indicates the IP protocol version of the MNP that follows. V is set to 0 for IPv4 or 1 for IPv6.

- FMT is a three bit code that determines the context and format of the L2ADDR exactly as specified in Figure 5.

- MNP{1], MNP[2], etc. up to MNP[i] encode the leading 'i' octets of the MNP, beginning with the most significant octet followed by the next most significant octet, etc. The number of MNP octets to be included is determined by the number of trailing zero octets in the prefix. For example, for the IPv6 MNP 2001:db8:1:2::/64, 'i' is set to 8 and only the leftmost 8 octets of the MNP are included. In the same way, for the IPv4 MNP 192.0.2/24, 'i' is set to 3 and only the leftmost 3 octets of the MNP are included.

- Link Layer Address (L2ADDR) is a UDP Port Number and IP address encoded according to FMT exactly as specified in Figure 5.

The node then forwards the packet via a local Bridge, which will eventually direct it to a Bridge on the same segment as the target.

When a Bridge receives a packet with Segments Left=1 and with LHS ID on a local segment, it checks to see if there is an AERO Route Optimization TLV. If so, the Bridge creates a ULA destination according to FMT. If FMT indicates that L2ADDR corresponds to a target Proxy/Server, the Bridge concatenates the SRT ::/16 prefix with the LHS ID to form the ULA destination. Otherwise, the Bridge concatenates the SRT ::/16 prefix with the leading MNPlen octets of the MNP and sets the remaining rightmost bits to 0 to form a Subnet Router Anycast ULA destination. The Bridge then writes the ULA into the SPAN header destination address and encapsulates the packet in an INET header with the target's L2ADDR as the destination then forwards the packet. Since the SRH contains no useful information for the destination, the Bridge may elect to delete the SRH before forwarding in order to reduce overhead.

In this way, the Bridge participates in route optimization to reduce traffic load and suboptimal routing through strict spanning tree paths. Note that if the Bridge does not recognize the AERO Route Optimization TLV, it instead places the LHS ID concatentaed with the SRT ::/16 prefix in the IPv6 destination address and forwards according to the spanning tree. (Note that this is the same behavior

that would occur if the AERO Route Optimization TLV were not present).

3.2.8. Segment Routing Header Compression

In the Segment Routing use cases discussed above, the segment routing headers must be kept to a minimum size since source and target Clients may be located behind low-end wireless links (e.g., 1Mbps or less). The Compressed Routing Header (CRH) [I-D.bonica-6man-comp-rtg-hdr] provides a compact form that reduces the header size by omitting invariant information. The CRH Helper option [I-D.bonica-6man-crh-helper-opt] can be used to encode the AERO Route Optimization TLV, and the final hop Bridge that performs route optimization may remove the CRH and its helper before encapsulating and forwarding to the target.

The CRH and its companion helper option are therefore seen as critical architectural elements that should be quickly progressed through the standards process. Implementations SHOULD use the CRH and its companion helper option instead of other Routing Header types whenever possible to conserve bandwidth.

3.3. OMNI Interface Characteristics

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

- INET interfaces connect to an INET either natively or through one or several IPv4 Network Address Translators (NATs). Native INET interfaces have global IP addresses that are reachable from any INET correspondent. All Server, Relay and Bridge interfaces are native interfaces, as are INET-facing interfaces of Proxys. NATed INET interfaces connect to a private network behind one or more NATs that provide INET access. Clients that are behind a NAT are required to send periodic keepalive messages to keep NAT state alive when there are no data packets flowing.

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

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

- Direct interfaces connect a Client directly to a Server or Proxy without crossing any ANET/INET paths. An example is a line-of-

sight link between a remote pilot and an unmanned aircraft. The same Client considerations apply as for VPNed interfaces.

- In all cases, Clients examine the P flag in received RAs. If the P flag is 1, the node that returned the RA message is acting as a Proxy; otherwise, it is acting as a Server.

OMNI interfaces use SPAN encapsulation as necessary as discussed in Section 3.2.4. OMNI interfaces use link-layer encapsulation (see: Section 3.6) to exchange packets with OMNI link neighbors over INET or VPNed interfaces. OMNI interfaces do not use link-layer encapsulation over Proxyed and Direct underlying interfaces.

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

OMNI interfaces send ND messages with an OMNI option formatted as specified in [I-D.templin-6man-omni-interface]. The OMNI option includes prefix registration information and "ifIndex-tuples" containing link information parameters for the OMNI interface's underlying interfaces.

SPAN-encapsulated OMNI interface ND messages also include a Source/Target Link-Layer Address Option (S/TLLAO) formatted as shown in Figure 5:

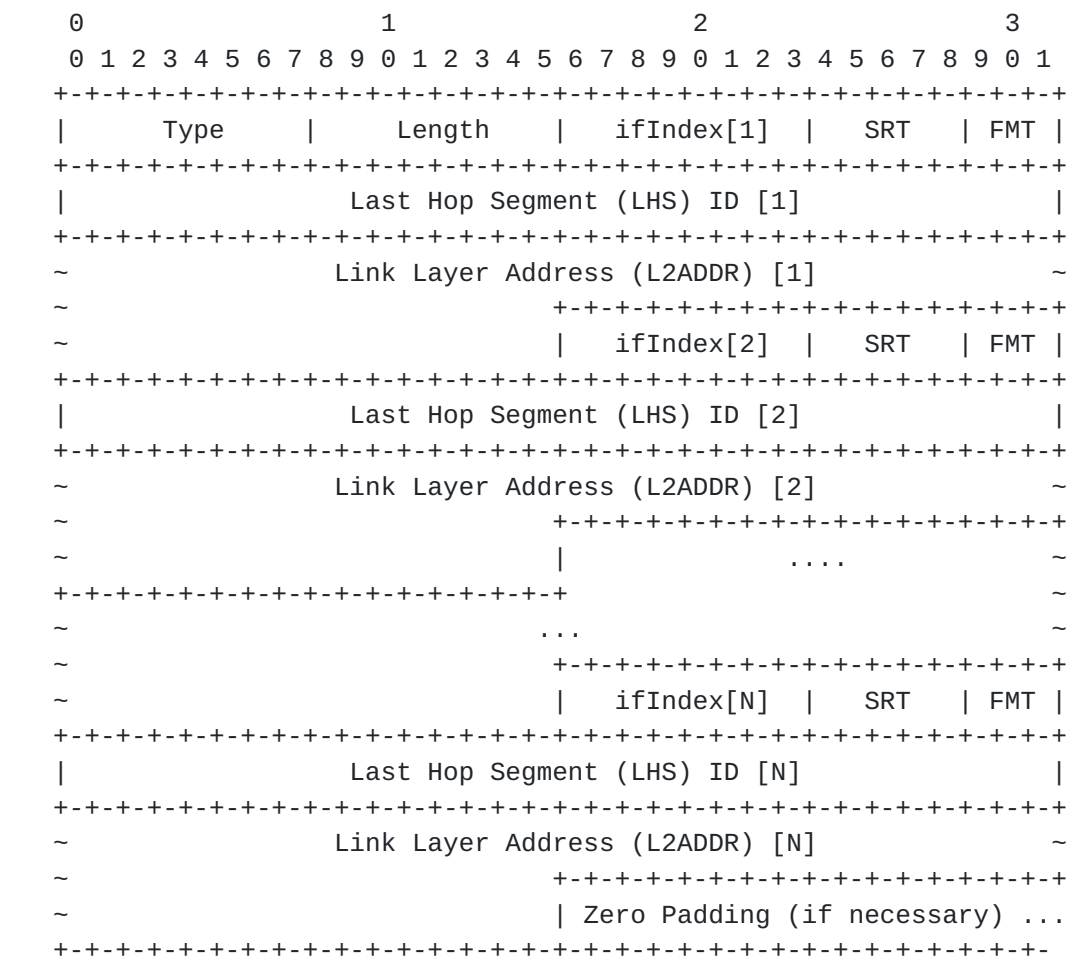

Figure 5: OMNI Source/Target Link-Layer Address Option (S/TLLAO) Format

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 corresponding to a proper subset of the ifIndex-tuples that appear in the OMNI option. Each ifIndex-tuple includes the following information:

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

- SRT - a 5-bit value that when added to 96 determines the prefix length to apply to the ULA formed from concatenating the SRT ::/16 prefix with Last Hop Segment (LHS) ID. For example, the prefix length for the value 16 is 112.

- FMT - a 3-bit "Framework/Mode/Type" code corresponding to the included Link Layer Address (L2ADDR) as follows:

o

  * When the most significant bit (i.e., "Framework") is set to 0, L2ADDR is the INET encapsulation address of a Proxy/Server; otherwise, it is the addresss for the Source/Target itself

  * When the next most significant bit (i.e., "Mode") is set to 0, the Source/Target L2ADDR is on the open INET; otherwise, it is (likely) located behind a NAT.

  * When the least significant bit (i.e., "Type") is set to 0, L2ADDR is an IPv4 address; else, it is an IPv6 address.

o Last Hop Segment (LHS) ID - Includes the least significant 32 bits of the last hop Proxy/Server ULA prior to encapsulation according to L2ADDR. When SRT and LHS are both set to 0, the last hop Proxy/Server ULA is considered unspecified in this IPv6 ND message.

o Link Layer Address (L2ADDR) - Included according to FMT, and identifies the link-layer address (i.e., the encapsulation address) of the source/target. The Port Number and IP address are recorded in ones-compliment "obfuscated" form per [RFC4380].

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

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

appear to have a single interface but with a dynamically changing link-layer address.

If the Client has multiple active underlying interfaces, then from the perspective of ND it would appear to have multiple link-layer addresses. In that case, ND message OMNI options MAY include multiple ifIndex-tuples - each with values that correspond 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.

Bridge, Server and Proxy OMNI interfaces may be configured over one or more secured tunnel interfaces. The OMNI interface configures both an LLA and its corresponding ULA, while the underlying secured tunnel interfaces are either unnumbered or configure the same ULA. The OMNI interface encapsulates each IP packet in a SPAN header and presents the packet to the underlying secured tunnel interface. Routing protocols such as BGP that run over the OMNI interface do not employ SPAN encapsulation, but rather present the routing protocol messages directly to the underlying secured tunnels while using the ULA as the source address. This distinction must be honored consistently according to each node's configuration so that the IP forwarding table will associate discovered IP routes with the correct interface.

3.4. OMNI Interface Initialization

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

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

3.4.1. AERO Server/Relay Behavior

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

The OMNI interface provides a single interface abstraction to the IP layer, but internally comprises multiple secured tunnels as well as an NBMA nexus for sending encapsulated data packets to OMNI 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.

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

3.4.2. AERO Proxy Behavior

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

3.4.3. AERO Client Behavior

When a Client enables an OMNI interface, it sends RS messages with ND/PD parameters over its underlying interfaces 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, or through one or more NATs in the case of a Client's INET interface.)

3.4.4. AERO Bridge Behavior

AERO Bridges configure an OMNI interface and assign the ULA Subnet Router Anycast address for each OMNI link segment they connect to. Bridges configure secured tunnels with Servers, Proxys and other Bridges; they also configure LLAs/ULAs and permanent neighbor cache entries the same as Servers. Bridges engage in a BGP routing protocol session with a subset of the Servers and other Bridges on the spanning tree (see: Section 3.2.3).

3.5. OMNI Interface Neighbor Cache Maintenance

Each OMNI interface maintains a conceptual neighbor cache that includes an entry for each neighbor it communicates with on the OMNI link per [RFC4861]. OMNI 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 Bridges maintain permanent neighbor cache entries for their associated Proxys and Servers (and vice-versa). Each entry maintains the mapping between the neighbor's network-layer LLA and corresponding INET address.

Symmetric neighbor cache entries are created and maintained through RS/RA exchanges as specified in Section 3.12, 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 OMNI 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 3.14, and are garbage-collected when keepalive timers expire. AERO 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 ROR) 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(s).

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

When an ROR receives an authentic NS message used for route optimization, it searches for a symmetric neighbor cache entry for the target Client. The ROR 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 REPORT_TIME seconds. The ROR resets ReportTime when it receives a new authentic NS message, and otherwise decrements ReportTime while no authentic NS messages have been received. It is RECOMMENDED that REPORT_TIME be set to the default constant value REACHABLE_TIME plus 10 seconds (40 seconds by default) to allow a window for route optimization to converge before ReportTime decrements below REACHABLE_TIME.

When the ROS receives a solicited NA message response to its NS message used for route optimization, it creates or updates 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 REACHABLE_TIME 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 REACHABLE_TIME be set to the default constant value 30 seconds as specified in [RFC4861].

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

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

3.6. OMNI Interface Encapsulation and Re-encapsulation

OMNI interfaces insert a mid-layer IPv6 header known as the SPAN header when necessary as discussed in the following sections. After either inserting or omitting the SPAN header, the OMNI interface also inserts or omits an outer encapsulation header as discussed below.

OMNI interfaces avoid outer encapsulation over Direct underlying interfaces and Proxyed underlying interfaces for which the first-hop access router is AERO-aware. Other OMNI interfaces encapsulate packets according to whether they are entering the OMNI interface

from the network layer or if they are being re-admitted into the same OMNI link they arrived on. This latter form of encapsulation is known as "re-encapsulation".

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

For packets undergoing re-encapsulation, the OMNI 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.)

OMNI interfaces configured over INET underlying interfaces encapsulate packets in INET headers according to the next hop determined in the forwarding algorithm in Section 3.10. If the next hop is reached via a secured tunnel, the OMNI interface uses an encapsulation format specific to the secured tunnel type (see: Section 6). If the next hop is reached via an unsecured INET interface, the OMNI interface instead uses UDP/IP encapsulation per [RFC4380] and as extended in [RFC6081].

When UDP/IP encapsulation is used, the OMNI interface next sets the UDP source port to a constant value that it will use in each successive packet it sends, and sets the UDP length field to the length of the encapsulated packet plus 8 bytes for the UDP header itself plus the length of any included extension headers or trailers. The encapsulated packet may be either IPv6 or IPv4, as distinguished by the version number found in the first four bits.

For UDP/IP-encapsulated packets sent to a Server, Relay or Bridge, the OMNI interface sets the UDP destination port to 8060, i.e., the IANA-registered port number for AERO. For packets sent to a Client, the OMNI interface sets the UDP destination port to the port value stored in the neighbor cache entry for this Client. The OMNI interface finally includes/omits the UDP checksum according to [RFC6935][RFC6936].

3.7. OMNI Interface Decapsulation

OMNI interfaces decapsulate packets destined either to the AERO node itself or to a destination reached via an interface other than the OMNI interface the packet was received on. When the encapsulated packet arrives in multiple SPAN fragments, the OMNI interface reassembles as discussed in Section 3.9. Further decapsulation steps are performed according to the appropriate encapsulation format specification.

3.8. OMNI Interface Data Origin Authentication

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

- AERO Bridges, Servers and Proxys accept encapsulated data packets and control messages received from the (secured) spanning tree.
- AERO Proxys and Clients accept packets that originate from within the same secured ANET.
- AERO Clients and Relays accept packets from downstream network correspondents based on ingress filtering.
- AERO Clients, Relays and Servers verify the outer UDP/IP encapsulation addresses according to [RFC4380].

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

3.9. OMNI Interface MTU and Fragmentation

The OMNI interface observes the link nature of tunnels, including the Maximum Transmission Unit (MTU) and the role of fragmentation and reassembly[I-D.ietf-intarea-tunnels]. OMNI interface MTU and fragmentation/reassembly procedures are specified in [I-D.templin-6man-omni-interface].

3.10. OMNI Interface Forwarding Algorithm

IP packets enter a node's OMNI 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 OMNI interface neighbor). All packets entering a node's OMNI interface first undergo data origin authentication as discussed in Section 3.8. Packets that satisfy data origin authentication are processed further, while all others

are dropped silently. OMNI interfaces wrap accepted packets in a SPAN header and SRH if necessary as discussed above.

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

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

If multiple outgoing interfaces and/or neighbor interfaces have a preference of "high", the AERO node replicates the 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 OMNI interface forwarding algorithms for Clients, Proxys, Servers and Bridges. 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 LLA).

3.10.1. Client Forwarding Algorithm

When an IP packet enters a Client's OMNI 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 encapsulates the packet in a SPAN header and SRH if necessary and fragments according to MTU requirements (see: Section 3.9).

When an IP packet enters a Client's OMNI 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 inner 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 3.11).

3.10.2. Proxy Forwarding Algorithm

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

When the Proxy receives a data packet from a Client within the ANET, the Proxy reassembles and re-fragments if necessary then searches for an asymmetric neighbor cache entry that matches the destination and forwards 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 using SPAN encapsulation and including a SRH if necessary according to the cached TLLAO information. 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 Bridge.

- else, the Proxy uses SPAN encapsulation and forwards the packet to a Bridge while using the ULA corresponding to the packet's destination as the SPAN destination address.

When the Proxy receives an encapsulated data packet from an INET neighbor or from a secured tunnel from a Bridge, 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, the Proxy reassembles the packet (if necessary) and continues processing.

Next if reassembly is complete and the neighbor cache state is REACHABLE, the Proxy returns a PTB if necessary (see: Section 3.9) then either drops or forwards the packet to the Client while performing SPAN 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 Bridge while performing re-fragmentation to 1280 bytes if necessary.

3.10.3. Server/Relay Forwarding Algorithm

For control messages destined to a target Client's LLA 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 3.16.

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. The Server first reassembles (if necessary) and forwards the packet (while re-fragmenting if necessary) as specified in Section 3.9.

- 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 ULA 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 OMNI link segment or using SPAN encapsulation and Segment Routing if necessary with the final destination set to the neighbor's ULA otherwise.

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

- else, the Server (acting as a Relay) 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 OMNI 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 OMNI interface. If the OMNI 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 OMNI interface itself.)

When the Server's OMNI interface receives a data packet from the network layer or from a VPNed or Direct Client, it performs SPAN encapsulation 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 Bridge using its own ULA as the source and the ULA corresponding to the destination as the destination.

3.10.4. Bridge Forwarding Algorithm

Bridges forward SPAN-encapsulated packets over secured tunnels the same as any IP router. When the Bridge receives a SPAN-encapsulated packet via a secured tunnel, it removes the outer INET header and searches for a forwarding table entry that matches the SPAN destination address. The Bridge then processes the packet as follows:

- if the destination matches its ULA Subnet Router Anycast address, the Bridge checks for a SRH. If there is a SRH with Segments Left=1, with the ULA of a Proxy/Server on the local segment as the LHS ID, and with an AERO Route Optimization TLV, the Bridge examines the FMT to determine if the target is behind a NAT. If no NAT is indicated, the Bridge copies the MNP Subnet Router Anycast address if an MNP is included (otherwise copies the Proxy/Server ULA) into the destination address then forwards the packet directly to the L2ADDR using link-layer (UDP/IP) encapsulation. If a NAT is indicated, the Bridge MAY perform NAT traversal procedures by sending bubbles per [RFC4380]. The Bridge then either applies AERO route optimization if NAT traversal procedures have been successfully applied, or forwards the packet directly to the Server.

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

- else, if the destination matches a forwarding table entry the Bridge forwards the packet via a secured tunnel to the next hop. If the destination matches an MSP without matching an MNP, however, the Bridge instead drops the packet and returns an ICMP

Destination Unreachable message subject to rate limiting (see: Section 3.11).

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

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

3.11. OMNI Interface Error Handling

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

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

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

The ICMP header is followed by the leading portion of the 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 6 (where, "L2" and "L3" refer to link-layer and network-layer, respectively):

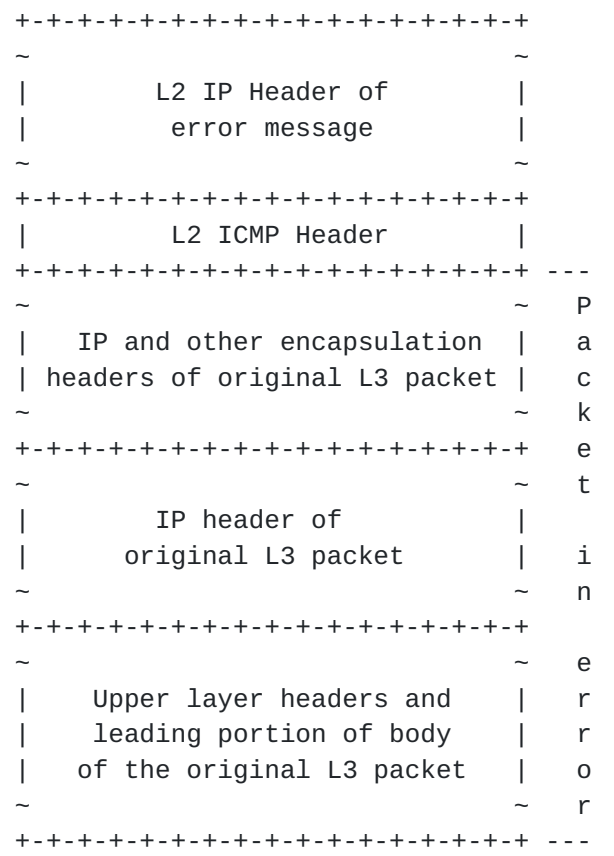

Figure 6: OMNI Interface Link-Layer Error Message Format

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 3.16.5.

- 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 Bridge 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 Bridge drops the packet and returns a network-layer Destination Unreachable message subject to rate limiting. The Bridge 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.

3.12. AERO Router Discovery, Prefix Delegation and Autoconfiguration

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

3.12.1. AERO ND/PD Service Model

Each AERO Server on the OMNI 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 OMNI 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 OMNI interfaces as advertising NBMA interfaces, and therefore send unicast RA messages with a short Router Lifetime value (e.g., ReachableTime seconds) in response to a Client's RS message. Thereafter, Clients send additional RS messages to keep 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 LLA 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.

3.12.2. 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 OMNI 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 LLA, it includes the LLA as the network-layer destination address; otherwise, it includes (link-local) All-Routers multicast as the

network-layer destination. If the Client already knows its own LLA, it uses the LLA 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 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 a VPN for VPNed interfaces, via a Proxy for proxyed interfaces or via INET encapsulation for INET interfaces) and waits for an RA message reply (see Section 3.12.3). The Client retries up to MAX_RTR_SOLICITATIONS times until an RA is received. If the Client receives no RAs, or if it receives an RA with Router Lifetime set to 0, the Client SHOULD abandon this 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 LLA 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 LLAs for each of the delegated MNPs and assigns them to the OMNI interface. The Client also caches any MSPs included in Route Information Options (RIOs) [RFC4191] as MSPs to associate with the OMNI link, and assigns the MTU value in the MTU option to the underlying interface.

The Client then registers additional underlying interfaces with the 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 LLA.

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.

3.12.3. AERO Server Behavior

AERO Servers act as IP routers and support a PD service for Clients. Servers arrange to add their LLAs 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 OMNI link.

When a Server receives a prospective Client's RS message on its OMNI interface, it SHOULD return an immediate RA reply with Router Lifetime set to 0 if it is currently too busy or otherwise unable to service the Client. Otherwise, the 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 3.2.3). 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 LLA 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.

Next, the Server prepares an RA message using its LLA 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 OMNI 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 OMNI link, plus an MTU option (see Section 3.9). 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 ReachableTime 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 sets the symmetric neighbor cache entry state to STALE and sets a 10 second timer. If the Server has not received a new RS or ND message with PD release indication before the 10 second timer expires, it deletes the neighbor cache entry and withdraws the MNP from the routing system.

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

3.12.3.1. 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 OMNI 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 OMNI 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.

3.13. The AERO Proxy

Clients may connect to protected-spectrum ANETs that deploy physical and/or link-layer 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 LLA, the Proxy proceeds to the next step. Otherwise, if the destination is (link-local) All-Routers multicast, the Proxy selects a "nearby" Server that is likely to be a good candidate to serve the Client and replaces the destination address with the Server's LLA. 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 ULA and destination set to the Server's ULA 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 ULA as the link-layer address. The Server then sends an RA message back to the Proxy via the spanning tree.

- 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 link-layer 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 Bridge using SPAN encapsulation with its own ULA as the source and the ULA corresponding to the Client as the destination. The Proxy instead forwards any Client data destined to an asymmetric neighbor cache target directly to the target according to the SPAN/

link-layer information - the process of establishing asymmetric neighbor cache entries is specified in Section 3.14.

While the Client is still attached to the ANET, the Proxy sends NS, RS and/or unsolicited NA messages to update the 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 (link-local) All-Nodes multicast 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 processing it locally.

If the Client becomes unreachable, the Proxy sets the neighbor cache entry state to DEPARTED and retains the entry for DepartTime seconds. While the state is DEPARTED, the Proxy forwards any packets destined to the Client to a Bridge via SPAN encapsulation with the Client's current Server as the destination. The Bridge 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 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 LLA and with destination address set to the LLA of the Client's selected Server or to (link-local) All-Routers multicast. The Client includes an OMNI option formatted as specified in [I-D.templin-6man-omni-interface].

The Client then sends the unencapsulated RS message, which will be intercepted by the AERO-Aware access router. The access router then

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.

3.13.1. Ancillary Servers Acting as Proxies

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

If some Client interfaces connect via different INET partitions, however, the Client still selects a single controlling Server and sends RS messages over interfaces that connect via ancillary Servers while using the LLA of the controlling Server as the destination.

When an ancillary Server receives an RS with destination set to the LLA of the controlling Server, it acts as a Proxy to forward the message to the controlling Server while forwarding the corresponding RA reply to the Client. When the ancillary Server forwards the RA reply, it sets the P bit in the RA flags field to indicate that it is acting in Proxy mode on behalf of this Client.

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

3.13.3. Point-to-Multipoint Server Coordination

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

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

Clients can thereafter employ efficient point-to-multipoint Server coordination under the assistance of the Proxy to 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 IPv6 ND messages to request the Proxy to release from former Servers via the procedures discussed in Section 3.16.5.

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

3.14. AERO Route Optimization / Address Resolution

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

The route optimization procedure is conducted between the ROS and the target Server/Relay 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 Relay.

The procedures are specified in the following sections.

3.14.1. 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 LLA 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 ULA and destination set to the ULA corresponding to the packet's final destination, then sends the message into the spanning tree without decrementing the network-layer TTL/Hop Limit field.

3.14.2. Relaying the NS

When the Bridge 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 Bridge then forwards the message toward the ROR via the spanning tree the same as for any IPv6 router. The final-hop Bridge in the spanning tree will deliver the message via a secured tunnel to the ROR.

3.14.3. 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 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 ULA of the Client's new Server, forwards the message into the spanning tree 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 LLA corresponding to the target, sets the Target Address 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.

The ROR sets L2ADDR to its own INET address for VPNed or Direct interfaces, to the INET address of the Proxy for Proxyed interfaces or to the Client's INET address for INET interfaces. The ROR then includes the lower 32 bits of its own ULA (or the ULA of the Proxy, for Proxyed interfaces) as the LHS ID, encodes the ULA prefix length code in the SRT field and sets the FMT code accordingly as specified in Section 3.3.

The ROR then sets the NA 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 ULA and destination set to the source ULA of the NS(AR) message, then forwards the message into the spanning tree without decrementing the network-layer TTL/Hop Limit field.

3.14.4. Relaying the NA

When the Bridge 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 Bridge then forwards the SPAN-encapsulated NA message toward the ROS the same as for any IPv6 router. The final-hop Bridge in the spanning tree will deliver the message via a secured tunnel to the ROS.

3.14.5. 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 ULA 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.

3.14.6. 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 A.1). The ROS uses the cached ULA 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 3.14.1 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 3.16). 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.

3.15. Neighbor Unreachability Detection (NUD)

AERO nodes perform Neighbor Unreachability Detection (NUD) per [RFC4861] either reactively in response to persistent link-layer errors (see Section 3.11) or proactively to confirm reachability. 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 Relays can use standard NS/NA NUD exchanges sent over the spanning tree 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.

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 spanning tree, 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 LLA as both the IPv6 source and destination address, and the MNP Subnet-Router anycast address as the Target Address. The ROS includes a Nonce and Timestamp option, then encapsulates the message in SPAN/INET headers with its own ULA as the source and the ULA of the route optimization target as the destination. The ROS then forwards the message to the target (either directly to the L2ADDR of the target if the target is in the same OMNI link segment, or via a Bridge if the target is in a different OMNI link 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 spanning tree). The route optimization target does not decrement the NS(NUD) message IPv6 Hop-Limit in the process, since the message has not exited the OMNI link.

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 an 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 OMNI interface forwarding algorithm specified in Section 3.10.

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

3.16. 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 OMNI link, as opposed to a Centralized Mobility Management (CMM) service where there is a single network mobility collective entity for all Clients. Clients coordinate with their associated 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.

3.16.1. 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 LLA, sets the destination address to (link-local) All-Nodes multicast and sets the Target Address to the Client's Subnet-Router anycast address. The Server also includes an OMNI option with prefix registration information and with ifIndex-tuples for the target Client's remaining interfaces. The Server then includes a TLLAO with corresponding ifIndex-tuples prepared the same as for the initial route optimization event. 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 ULA and destination set to the ULA of the ROS and sends the message into the spanning tree.

As discussed in Section 7.2.6 of [RFC4861], the transmission and reception of uNA messages is unreliable but provides a useful

optimization. In well-connected Internetworks with robust data links uNA messages will be delivered with high probability, but in any case the Server can optionally send up to MAX_NEIGHBOR_ADVERTISEMENT uNAs to each ROS to increase the likelihood that at least one will be received.

When the ROS receives a uNA message, 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.

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

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

Up to MAX_RTR_SOLICITATIONS 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.

3.16.3. Bringing New Links Into Service

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

3.16.4. 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.

3.16.5. Moving to a New Server

When a Client associates with a new Server, it performs the Client procedures specified in Section 3.12.2. 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 3.12.3 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 LLA as the source address, the old Server's LLA 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 ULA as the source and the old Server's ULA as the destination, then sends the message into the spanning tree.

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 ULA, and resets DepartTime. 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 with the ULA 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 for a period of 10 seconds after which the cache entry is deleted. While in the STALE state, subsequent data packets 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 OMNI link 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.

3.17. Multicast

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

3.17.1. Source-Specific Multicast (SSM)

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

For each S belonging to a prefix reachable via X's OMNI interface, X originates a separate copy of the Join/Prune for each (S,G) in the

message using its own LLA as the source address and ALL-PIM-ROUTERS as the destination address. X then encapsulates each message in a SPAN header with source address set to the ULA of X and destination address set to S then forwards the message into the spanning tree, which delivers it to AERO Server/Relay "Y" that services S. At the same time, if the message was a Join, X sends a route-optimization NS message toward each S the same as discussed in Section 3.14. The resulting NAs will return the LLA for the prefix that matches S as the network-layer source address and TLLAOs with the ULA corresponding to any ifIndex-tuples that are currently servicing S.

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

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 spanning tree.

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

After some later time, C may move to a new Server Y2 and depart from old Sever Y1. In that case, Y1 sends Join messages for any of C's

active (S,G) groups to Y2 while including its own LLA as the source address. This causes Y2 to include Y1 in the multicast forwarding tree during the interim time that Y1's 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 spanning tree and re-initiates route optimization the same as if it were receiving a fresh Join message from a node on a downstream link.

3.17.2. Any-Source Multicast (ASM)

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

For each source S that sends multicast traffic to group G via R, the Proxy/Server Z* for the Client that aggregates S encapsulates the packets in PIM Register messages and forwards them to R via the spanning tree, which may then elect to send a PIM Join to Z*. This will result in an (S,G) tree rooted at Z* with R as the next hop so that R will begin to receive two copies of the 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 3.17.1. 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.

3.17.3. Bi-Directional PIM (BIDIR-PIM)

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

3.18. Operation over Multiple OMNI Links

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

The Bridges, Servers and Proxys on each OMNI link can assign AERO and ULAs 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 LLA such as fe80::1000 could be used to number distinct nodes that connect to different OMNI links.

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

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

In a first alternative, if each OMNI link services different MSPs, then the Client can receive a distinct MNP from each of the links. IP routing will therefore assure that the correct 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 Bridges.

In a second alternative, if each OMNI link services the same MSP(s) then each link could assign a distinct "OMNI link Anycast" address that is configured by all Bridges on the link. Correspondent nodes

can then perform Segment Routing to select the correct SRT, which will then direct the packet over multiple hops to the target.

3.19. DNS Considerations

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

When an AERO Client registers with an AERO 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.

3.20. Transition Considerations

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

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

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

3.21. Detecting and Reacting to Server and Bridge Failures

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

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

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

3.22. AERO Clients on the Open Internet

AERO Clients that connect to the open Internet via INET interfaces can establish a VPN or direct link to securely connect to a Server in a "tethered" arrangement with all of the Client's traffic transiting the Server. Alternatively, the Client can associate with an INET Server using UDP/IP encapsulation and asymmetric securing services as discussed in the following sections.

When a Client's OMNI interface enables an INET underlying interface, it first determines whether the interface is likely to be behind a NAT. For IPv4, the Client assumes it is on the open Internet if the

INET address is not a special-use IPv4 address per [RFC3330]. Similarly for IPv6, the Client assumes it is on the open Internet if the INET address is not a link-local [RFC4291] or unique-local [RFC4193] IPv6 address.

The Client then prepares a UDP/IP-encapsulated RS message with IPv6 source address set to its LLA, with IPv6 destination set to (link-local) All-Routers multicast and with an OMNI option with underlying interface parameters. If the Client believes that it is on the open Internet, it SHOULD also include an SLLAO set according to the address used for INET encapsulation (otherwise, it MAY omit the SLLAO). If the underlying address is IPv4, the Client includes the Port Number and IPv4 address written in obfuscated form [RFC4380] as discussed in Section 3.3. If the underlying interface address is IPv6, the Client instead includes the Port Number and IPv6 address in obfuscated form. The Client finally includes an Authentication option per [RFC4380] to provide message authentication, sets the UDP/IP source to its INET address and UDP port, sets the UDP/IP destination to the Server's INET address and the AERO service port number (8060), then sends the message to the Server.

When the Server receives the RS, it authenticates the message and registers the Client's MNP and INET interface information according to the OMNI option parameters. If the RS message includes an SLLAO, the Server compares the encapsulation IP address and UDP port number with the (unobfuscated) SLLAO values. If the values are the same, the Server caches the Client's information as "INET" addresses meaning that the Client is likely to accept direct messages without requiring NAT traversal exchanges. If the values are different (or, if there was no SLLAO) the Server instead caches the Client's information as "NAT" addresses meaning that NAT traversal exchanges may be necessary.

The Server then returns an RA message with IPv6 source and destination set corresponding to the addresses in the RS, and with an Authentication option per [RFC4380]. For IPv4, the Server also includes an Origin option per [RFC4380] with the mapped and obfuscated Port Number and IPv4 address observed in the encapsulation headers. For IPv6, the Server instead includes an IPv6 Origin option per Figure 7 with the mapped and obfuscated observed Port Number and IPv6 address (note that the value 0x02 in the second octet differentiates from other [RFC4380] option types).

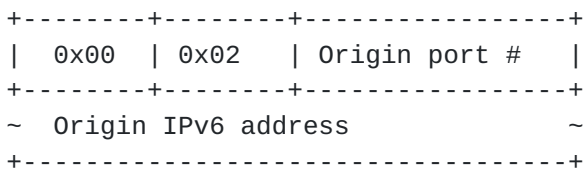

Figure 7: IPv6 Origin Option

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

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

When the Client sends messages to target IP addresses, it also invokes route optimization per Section 3.14 using IPv6 ND address resolution messaging. The Client sends the NS(AR) message to the Server wrapped in a UDP/IP header with an Authentication option with the NS source address set to the Client's LLA and destination address set to the target solicited node multicast address. The Server authenticates the message and sends a corresponding NS(AR) message over the spanning tree the same as if it were the ROS, but with the SPAN source address set to the Server's ULA and destination set to the ULA of the target. When the ROR receives the NS(AR), it adds the Server's ULA and Client's LLA to the target's Report List, and returns an NA with OMNI and TLLAO information for the target. The Server then returns a UDP/IP encapsulated NA message with an Authentication option to the Client.

Following route optimization, for targets in the same OMNI link segment if the target's TLLAO addresss is on the open INET, the Client forwards data packets directly to the target INET address. If the target's TLLAO address is behind a NAT, the Client first establishes NAT state for the L2ADDR using the "bubble" mechanisms specified in [RFC6081][RFC4380]. The Client continues to send data packets via its Server until NAT state is populated, then begins forwarding packets via the direct path through the NAT to the target. For targets in different OMNI link segments, the Client inserts an

SRH and forwards data packets to the Bridge that returned the NA message.

The ROR may return uNAs via the Server if the target moves, and the Server will send corresponding Authentication-protected uNAs to the Client. The Client can also send "loopback" NS(NUD) messages to test forward path reachability even though there is no security association between the Client and the target.

The Client sends UDP/IP encapsulated IPv6 packets no larger than 1280 bytes in one piece. In order to accommodate larger IPv6 packets (up to the OMNI interface MTU), the Client inserts a SPAN header with source set to its own ULA and destination set to the ULA of the target and uses IPv6 fragmentation according to Section 3.9. The Client then encapsulates each fragment in a UDP/IP header and sends the fragments to the next hop.

3.22.1. Use of SEND and CGA

In some environments, use of the [RFC4380] Authentication option alone may be sufficient for assuring IPv6 ND message authentication between Clients and Servers. When additional protection is necessary, nodes should employ SEcure Neighbor Discovery (SEND) [RFC3971] with Cryptographically-Generated Addresses (CGA) [RFC3972].

When SEND/CGA are used, the Client prepares RS messages with its link-local CGA as the IPv6 source and (link-local) All-Routers multicast as the IPv6 Destination, includes any SEND options and wraps the message in a SPAN header. The Client sets the SPAN source address to its own ULA and sets the SPAN destination address to (site-local) All-Routers multicast. The Client then wraps the RS message in UDP/IP headers according to [RFC4380] and sends the message to the Server.

When the Server receives the message, it first verifies the Authentication option (if present) then uses the SPAN source address to determine the MNP of the Client. The Server then processes the SEND options to authenticate the RS message and prepares an RA message response. The Server prepares the RA with its own link-local CGA as the IPv6 source and the CGA of the Client as the IPv6 destination, includes any SEND options and wraps the message in a SPAN header. The Server sets the SPAN source address to its own ULA and sets the SPAN destination address to the Client's ULA. The Server then wraps the RA message in UDP/IP headers according to [RFC4380] and sends the message to the Client. Thereafter, the Client/Server send additional RS/RA messages to maintain their association and any NAT state.

The Client and Server also may exchange NS/NA messages using their own CGA as the source and with SPAN encapsulation as above. When a Client sends an NS(AR), it sets the IPv6 source to its CGA and sets the IPv6 destination to the Solicited-Node Multicast address of the target. The Client then wraps the message in a SPAN header with its own ULA as the source and the ULA of the target as the destination and sends it to the Server. The Server authenticates the message, then changes the IPv6 source address to the Client's LLA, removes the SEND options, and sends a corresponding NS(AR) into the spanning tree. When the Server receives the corresponding SPAN-encapsulated NA, it changes the IPv6 destination address to the Client's CGA, inserts SEND options, then wraps the message in UDP/IP headers and sends it to the Client.

When a Client sends a uNA, it sets the IPv6 source address to its own CGA and sets the IPv6 destination address to (link-local) All-Nodes multicast, includes SEND options, wraps the message in SPAN and UDP/IP headers and sends the message to the Server. The Server authenticates the message, then changes the IPv6 address to the Client's LLA, removes the SEND options and forwards the message the same as discussed in Section 3.16.1. In the reverse direction, when the Server forwards a uNA to the Client, it changes the IPv6 address to its own CGA and inserts SEND options then forwards the message to the Client.

When a Client sends an NS(NUD), it sets both the IPv6 source and destination address to its own LLA, wraps the message in a SPAN header and UDP/IP headers, then sends the message directly to the peer which will loop the message back. In this case alone, the Client does not use the Server as a trust broker for forwarding the ND message.

3.23. Time-Varying MNPs

In some use cases, it is desirable, beneficial and efficient for the Client to receive a constant MNP that travels with the Client wherever it moves. For example, this would allow air traffic controllers to easily track aircraft, etc. In other cases, however (e.g., intelligent transportation systems), the MN may be willing to sacrifice a modicum of efficiency in order to have time-varying MNPs that can be changed every so often to defeat adversarial tracking.

The DHCPv6-PD service offers a way for Clients that desire time-varying MNPs to obtain short-lived prefixes (e.g., on the order of a small number of minutes). In that case, the identity of the Client would not be bound to the MNP but rather the Client's identity would be bound to the DHCPv6 Device Unique Identifier (DUID) and used as the seed for Prefix Delegation. The Client would then be obligated

to renumber its internal networks whenever its MNP (and therefore also its LLA) changes. This should not present a challenge for Clients with automated network renumbering services, however presents limits for the durations of ongoing sessions that would prefer to use a constant address.

4. Implementation Status

An 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.

As of 4/1/2020, more recent updated implementations are under internal development and testing with plans to release in the near future.

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.

The IANA is instructed to assign a new type value TBD in the Segment Routing Header TLV registry [RFC8754].

No further IANA actions are required.

6. Security Considerations

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

AERO Servers, Relays and Proxys targeted by a route optimization may also receive data packets directly from arbitrary nodes in INET partitions instead of via the spanning tree. For INET partitions that apply effective ingress filtering to defeat source address spoofing, the simple data origin authentication procedures in Section 3.8 can be applied.

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

AERO Clients that connect to secured ANETs need not apply security to their ND messages, since the messages will be intercepted by a perimeter Proxy that applies security on its INET-facing interface as part of the spanning tree (see above). AERO Clients connected to the open INET can use symmetric network and/or transport layer security services such as VPNs or can by some other means establish a direct link. When a VPN or direct link may be impractical, however, an asymmetric security service such as SEcure Neighbor Discovery (SEND) [RFC3971] with Cryptographically Generated Addresses (CGAs) [RFC3972] and/or the Authentication option [RFC4380] can be applied.

Application endpoints SHOULD use application-layer security services such as TLS/SSL, DTLS or SSH [RFC4251] to assure the same level of protection as for critical secured Internet services. AERO Clients that require host-based VPN services SHOULD use 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 Bridges present targets for traffic amplification Denial of Service (DoS) attacks. This concern is no different than for widely-deployed VPN security gateways in the Internet, where attackers could send spoofed packets to the gateways at high data rates. This can be mitigated by connecting Servers and Bridges over dedicated links with no connections to the Internet and/or when connections to the Internet are only permitted through well-managed firewalls. Traffic amplification DoS attacks can also target an AERO Client's low data rate links. This is a concern not only for Clients located on the open Internet but also for Clients in secured enclaves. AERO Servers and Proxys can institute rate limits that protect Clients from receiving packet floods that could DoS low data rate links.

AERO Relays must implement ingress filtering to avoid a spoofing attack in which spurious messages with ULA addresses are injected

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

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

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

SRH authentication facilities are specified in [RFC8754].

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

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

7. Acknowledgements

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

This work has further been encouraged and supported by Boeing colleagues including Kyle Bae, M. Wayne Benson, Dave Bernhardt, Cam Brodie, John Bush, Balaguruna Chidambaram, Irene Chin, Bruce Cornish,

Claudiu Danilov, Don Dillenburg, Joe Dudkowski, Wen Fang, Samad Farooqui, Anthony Gregory, Jeff Holland, Seth Jahne, Brian Jaury, Greg Kimberly, Ed King, Madhuri Madhava Badgandi, Laurel Matthew, Gene MacLean III, Rob Muszkiewicz, Sean O'Sullivan, Vijay Rajagopalan, Greg Saccone, Rod Santiago, Kent Shuey, Brian Skeen, Mike Slane, Carrie Spiker, Katie Tran, Brendan Williams, Amelia Wilson, Julie Wulff, Yueli Yang, Eric Yeh and other members of the Boeing mobility, networking and autonomy teams. Kyle Bae, Wayne Benson, Katie Tran and Eric Yeh are especially acknowledged for implementing the AERO functions as extensions to the public domain OpenVPN distribution.

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

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

- 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-25 (work in progress), June 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>.

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

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

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

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

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <https://www.rfc-editor.org/info/rfc4380>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

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

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

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

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

8.2. Informative References

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

[I-D.bonica-6man-comp-rtg-hdr] Bonica, R., Kamite, Y., Niwa, T., Alston, A., and L. Jalil, "The IPv6 Compact Routing Header (CRH)", draft-bonica-6man-comp-rtg-hdr-22 (work in progress), May 2020.

[I-D.bonica-6man-crh-helper-opt] Li, X., Bao, C., Ruan, E., and R. Bonica, "Compressed Routing Header (CRH) Helper Option", draft-bonica-6man-crh-helper-opt-01 (work in progress), May 2020.

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

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

[I-D.ietf-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-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>.

[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>.

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

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

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

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

[RFC3330] IANA, "Special-Use IPv4 Addresses", RFC 3330, DOI 10.17487/RFC3330, September 2002, <https://www.rfc-editor.org/info/rfc3330>.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

Appendix A. Non-Normative Considerations

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

A.1. Implementation Strategies for Route Optimization

Route optimization as discussed in Section 3.14 results in the route optimization source (ROS) creating 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 while data packets are still flowing. However, the decision of when to initiate a new NS/NA exchange and to perpetuate the process is left as an implementation detail.

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

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

A.2. Implicit Mobility Management

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

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

the Client's new address. This implicit mobility method applies to use cases such as cellphones with both WiFi and Cellular interfaces where only one of the interfaces is active at a given time, and the Client automatically switches over to the backup interface if the primary interface fails.

A.3. Direct Underlying Interfaces

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

A.4. AERO Critical Infrastructure Considerations

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

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

A.5. 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 ReachableTime 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.

A.6. AERO Client / Server Architecture

The AERO architectural model is client / server in the control plane, with route optimization in the data plane. The same as for common Internet services, the AERO Client discovers the addresses of AERO 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 LLAs 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 B. Change Log

<< RFC Editor - remove prior to publication >>

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

- Updates on Segment Routing and S/TLLAO contents.

- Various editorials and addressing cleanups.

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

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

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

- Truncated obsolete back section matter.

Author's Address

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

Email: fltemplin@acm.org