Network Working Group
Internet-Draft
Obsoletes: rfc6706 (if approved)
Intended status: Standards Track
Expires: February 27, 2015

F. Templin, Ed.
Boeing Research & Technology

August 26, 2014

Transmission of IP Packets over AERO Links
draft-templin-aerolink-32.txt

Abstract

This document specifies the operation of IP over tunnel virtual links using Asymmetric Extended Route Optimization (AERO). Nodes attached to AERO links can exchange packets via trusted intermediate routers that provide forwarding services to reach off-link destinations and redirection services for route optimization. AERO provides an IPv6 link-local address format known as the AERO address that supports operation of the IPv6 Neighbor Discovery (ND) protocol and links IPv6 ND to IP forwarding. Admission control and provisioning are supported by the Dynamic Host Configuration Protocol for IPv6 (DHCPv6), and node mobility is naturally supported through dynamic neighbor cache updates. Although DHCPv6 and IPv6 ND messaging is used in the control plane, both IPv4 and IPv6 are supported in the data plane.

Status of This Memo

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

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

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

This Internet-Draft will expire on February 27, 2015.

Table of Contents

1. Introduction

This document specifies the operation of IP over tunnel virtual links using Asymmetric Extended Route Optimization (AERO). The AERO link can be used for tunneling to neighboring nodes over either IPv6 or IPv4 networks, i.e., AERO views the IPv6 and IPv4 networks as equivalent links for tunneling. Nodes attached to AERO links can exchange packets via trusted intermediate routers that provide forwarding services to reach off-link destinations and redirection services for route optimization that addresses the requirements outlined in [RFC5522].

AERO provides an IPv6 link-local address format known as the AERO address that supports operation of the IPv6 Neighbor Discovery (ND) [RFC4861] protocol and links IPv6 ND to IP forwarding. Admission control and provisioning are supported by the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC3315], and node mobility is naturally supported through dynamic neighbor cache updates. Although DHCPv6 and IPv6 ND message signalling is used in the control plane, either of IPv4 and IPv6 can be used in the data plane. The remainder of this document presents the AERO specification.

2. Terminology

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

AERO link
: a Non-Broadcast, Multiple Access (NBMA) tunnel virtual overlay configured over a node's attached IPv6 and/or IPv4 networks. All nodes on the AERO link appear as single-hop neighbors from the perspective of the virtual overlay.

AERO interface
: a node's attachment to an AERO link.

AERO address
an IPv6 link-local address constructed as specified in Section 3.2 and applied to a Client's AERO interface.

AERO node
a node that is connected to an AERO link and that participates in IPv6 ND over the link.

AERO Client ("Client")
a node that applies an AERO address to an AERO interface and receives an IP prefix delegation.

AERO Server ("Server")
a node that configures an AERO interface to provide default forwarding and DHCPv6 services for AERO Clients. The Server applies the IPv6 link-local subnet router anycast address (fe80::) to the AERO interface and also applies an administratively assigned IPv6 link-local unicast address used for operation of the IPv6 ND protocol.

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

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

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

underlying network
a connected IPv6 or IPv4 network routing region over which the tunnel virtual overlay is configured. A typical example is an enterprise network.

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

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

addresses are used as the encapsulation header source and destination addresses.

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

end user network (EUN)
an internal virtual or external edge IP network that an AERO Client connects to the rest of the network via the AERO interface.

AERO Service Prefix (ASP)
an IP prefix associated with the AERO link and from which AERO Client Prefixes (ACPs) are derived (for example, the IPv6 ACP 2001:db8:1:2::/64 is derived from the IPv6 ASP 2001:db8::/32).

AERO Client Prefix (ACP)
a more-specific IP prefix taken from an ASP and delegated to a Client.

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

Throughout the document, it is said that an address is "applied" to an AERO interface since the address need not always be "assigned" to the interface in the traditional sense. However, the address must at least be bound to the interface in some fashion to support the operation of DHCPv6 and the IPv6 ND protocol.

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

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Asymmetric Extended Route Optimization (AERO)

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

3.1. AERO Link Reference Model

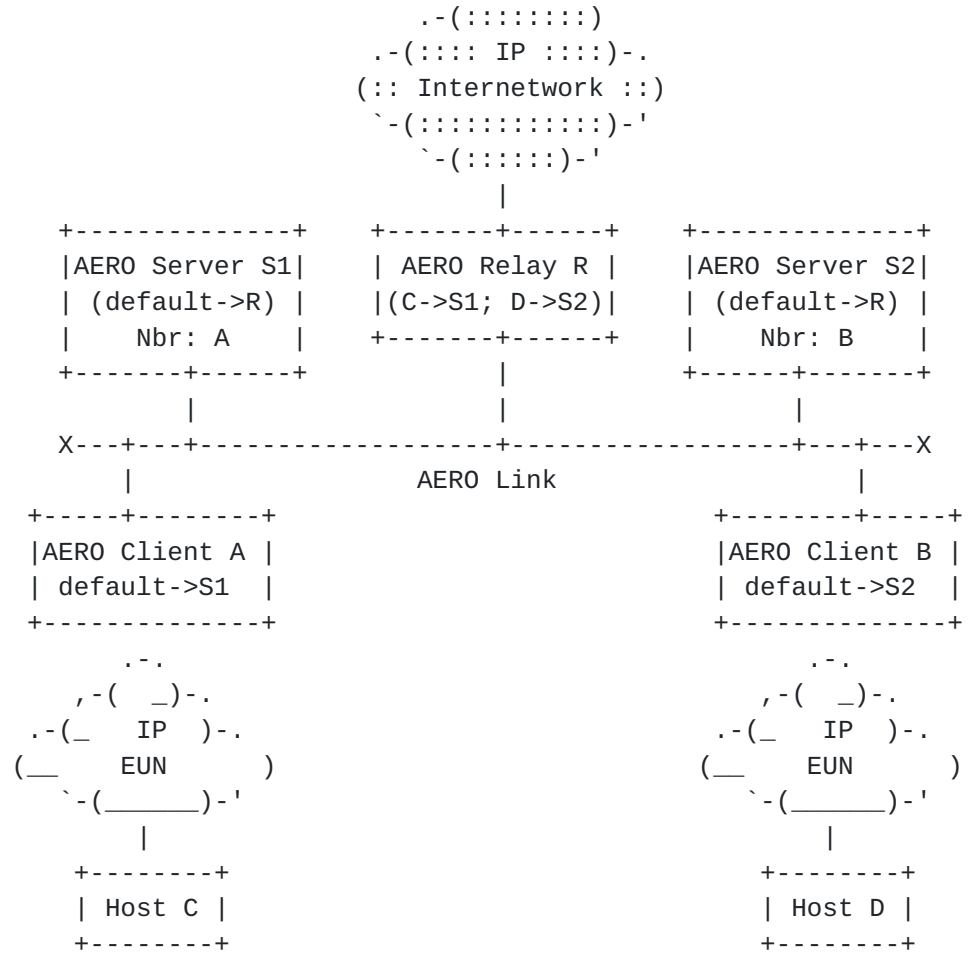

Figure 1: AERO Link Reference Model

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

- Relay R act as a default router for its associated Servers S1 and S2, and connects the AERO link to the rest of the IP Internetwork
- Servers S1 and S2 associate with Relay R and also act as default routers for their associated Clients A and B.
- Clients A and B associate with Servers S1 and S2, respectively and also act as default routers for their associated EUNs
- Hosts C and D attach to the EUNs served by Clients A and B, respectively

In operational practice, there may be many additional Relays, Servers and Clients.

3.2. AERO Node Types

AERO Relays provide default forwarding services to AERO Servers. Relays forward packets between Servers connected to the same AERO link and also forward packets between the AERO link and the native Internetwork. Relays present the AERO link to the native Internetwork as a set of one or more ASPs. Each Relay advertises the ASPs for the AERO link into the native IP Internetwork and serves as a gateway between the AERO link and the Internetwork. AERO Relays maintain an AERO interface neighbor cache entry for each AERO Server, and maintain an IP forwarding table entry for each AERO Client.

AERO Servers provide default forwarding services to AERO Clients. Each Server also peers with each Relay in a dynamic routing protocol session to advertise its list of associated Clients. Servers configure a DHCPv6 server function to facilitate Prefix Delegation (PD) exchanges with Clients. Each delegated prefix becomes an AERO Client Prefix (ACP) taken from an ASP. Servers forward packets between Clients and Relays, as well as between Clients and other Clients associated with the same Server. AERO Servers maintain an AERO interface neighbor cache entry for each AERO Relay. They also maintain both a neighbor cache entry and an IP forwarding table entry for each of their associated Clients.

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

AERO Clients that act as routers sub-delegate portions of their ACPs to links on EUNs. End system applications on Clients that act as routers bind to EUN interfaces (i.e., and not the AERO interface).

AERO Clients that act as ordinary hosts assign one or more IP addresses from their ACPs to the AERO interface but DO NOT assign the ACP itself to the AERO interface. Instead, the Client assigns the ACP to a "black hole" route so that unused portions of the prefix are nullified. End system applications on Clients that act as hosts bind directly to the AERO interface.

3.3. AERO Addresses

An AERO address is an IPv6 link-local address with an embedded ACP and applied to a Client's AERO interface. The AERO address is formed as follows:

fe80::[ACP]

For IPv6, the AERO address begins with the prefix fe80::/64 and includes in its interface identifier the base prefix taken from the Client's IPv6 ACP. The base prefix is determined by masking the ACP with the prefix length. For example, if the AERO Client receives the IPv6 ACP:

2001:db8:1000:2000::/56

it constructs its AERO address as:

fe80::2001:db8:1000:2000

For IPv4, the AERO address is formed from the lower 64 bits of an IPv4-mapped IPv6 address [RFC4291] that includes the base prefix taken from the Client's IPv4 ACP. For example, if the AERO Client receives the IPv4 ACP:

192.0.2.32/28

it constructs its AERO address as:

fe80::FFFF:192.0.2.32

The AERO address remains stable as the Client moves between topological locations, i.e., even if its link-layer addresses change.

NOTE: In some cases, prospective neighbors may not have a priori knowledge of the Client's ACP length and may therefore send initial IPv6 ND messages with an AERO destination address that matches the ACP but does not correspond to the base prefix. In that case, the Client MUST accept the address as equivalent to the base address, but then use the base address as the source address of any IPv6 ND message replies. For example, if the Client receives the IPv6 ACP 2001:db8:1000:2000::/56 then subsequently receives an IPv6 ND message with destination address fe80::2001:db8:1000:2001, it accepts the message but uses fe80::2001:db8:1000:2000 as the source address of any IPv6 ND replies.

3.4. AERO Interface Characteristics

AERO interfaces use IP-in-IPv6 encapsulation [RFC2473] to exchange tunneled packets with AERO neighbors attached to an underlying IPv6 network, and use IP-in-IPv4 encapsulation [RFC2003][RFC4213] to exchange tunneled packets with AERO neighbors attached to an underlying IPv4 network. AERO interfaces can also coordinate secured tunnel types such as IPsec [RFC4301] or TLS [RFC5246]. When Network Address Translator (NAT) traversal and/or filtering middlebox traversal may be necessary, a UDP header is further inserted immediately above the IP encapsulation header.

AERO interfaces maintain a neighbor cache, and AERO Clients and Servers use an adaptation of standard unicast IPv6 ND messaging. AERO interfaces use unicast Neighbor Solicitation (NS), Neighbor Advertisement (NA), Router Solicitation (RS) and Router Advertisement (RA) messages the same as for any IPv6 link. AERO interfaces use two redirection message types -- the first known as a Predirect message and the second being the standard Redirect message (see Section 3.9). AERO links further use link-local-only addressing; hence, AERO nodes ignore any Prefix Information Options (PIOs) they may receive in RA messages.

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

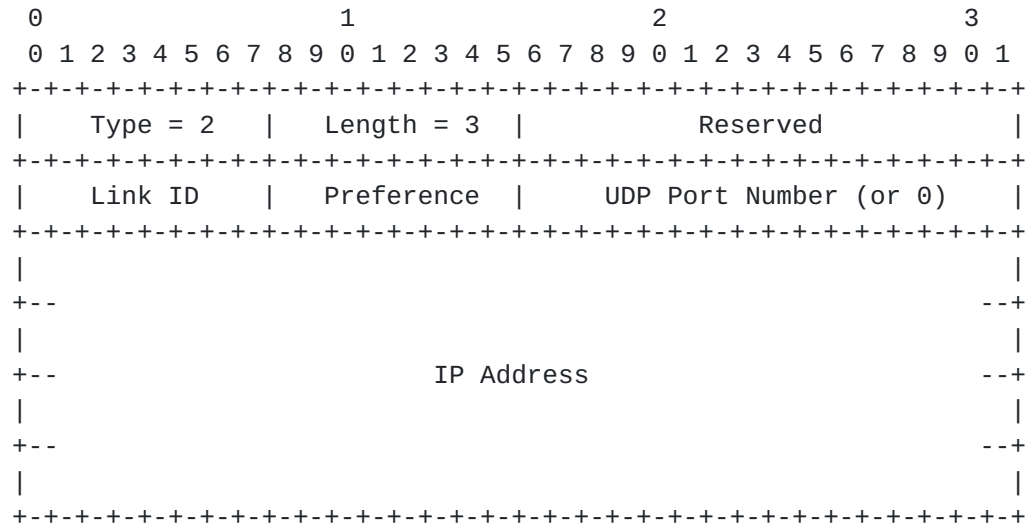

Figure 2: AERO Target Link-Layer Address Option (TLLAO) Format

In this format, Link ID is an integer value between 0 and 255 corresponding to an underlying interface of the target node, and Preference is an integer value between 0 and 255 indicating the

node's preference for this underlying interface (with 255 being the highest preference, 1 being the lowest and 0 meaning "link disabled"). UDP Port Number and IP Address are set to the addresses used by the target node when it sends encapsulated packets over the underlying interface. When no UDP encapsulation is used, UDP Port Number is set to 0. When the encapsulation IP address family is IPv4, IP Address is formed as an IPv4-mapped IPv6 address [RFC4291].

When a Relay enables an AERO interface, it applies an administratively assigned link-local address fe80::ID to the interface for communicating with Servers on the link. Each fe80::ID address MUST be unique among all Relays and Servers on the link, and MUST NOT collide with any potential AERO addresses, e.g., the addresses could be assigned as fe80::1, fe80::2, fe80::3, etc. The Relay also maintains an IP forwarding table entry for each Client-Server association and maintains a neighbor cache entry for each Server on the link. Relays do not require the use of IPv6 ND messaging for reachability determination since Relays and Servers engage in a dynamic routing protocol over the AERO interface. At a minimum, however, Relays respond to NS messages by returning an NA.

When a Server enables an AERO interface, it applies the address fe80:: to the interface as a link-local Subnet Router Anycast address, and also applies an administratively assigned link-local address fe80::ID to support the operation of DHCPv6 and the IPv6 ND protocol as well as to communicate with Relays on the link. The Server configures a DHCPv6 server function to facilitate DHCPv6 PD exchanges with AERO Clients. The Server also maintains a neighbor cache entry for each Relay on the link, and manages per-Client neighbor cache entries and IP forwarding table entries based on DHCPv6 exchanges. When the Server receives an NS/RS message on the AERO interface it returns an NA/RA message but does not update the neighbor cache. Servers also engage in a dynamic routing protocol with all Relays on the link. Finally, the Server provides a simple conduit between Clients and Relays, or between Clients and other Clients. Therefore, packets enter the Server's AERO interface from the link layer and are forwarded back out the link layer without ever leaving the AERO interface and therefore without ever disturbing the network layer.

When a Client enables an AERO interface, it invokes DHCPv6 PD to receive an ACP from an AERO Server. Next, it applies the corresponding AERO address to the AERO interface and creates a neighbor cache entry for the Server, i.e., the PD exchange bootstraps the provisioning of a unique link-local address. The Client maintains a neighbor cache entry for each of its Servers and each of its active peer Clients. When the Client receives Redirect/Predirect messages on the AERO interface it updates or creates neighbor cache

entries, including link-layer address information. Unsolicited NA messages update the cached link-layer addresses for the neighbor Client (e.g., following a link-layer address change due to node mobility) but do not create new neighbor cache entries. NS/NA messages used for Neighbor Unreachability Detection (NUD) update timers in existing neighbor cache entires but do not update link-layer addresses nor create new neighbor cache entries. Finally, the Client need not maintain any IP forwarding table entries for neighboring Clients. Instead, it can set a single "route-to-interface" default route in the IP forwarding table pointing to the AERO interface, and all forwarding decisions can be made within the AERO interface based on neighbor cache entries.

3.4.1. Coordination of Multiple Underlying Interfaces

AERO interfaces may be configured over multiple underlying interfaces. For example, common mobile handheld devices have both wireless local area network ("WLAN") and cellular wireless links. These links are typically used "one at a time" with low-cost WLAN preferred and highly-available cellular wireless as a standby. In a more complex example, aircraft frequently have many wireless data link types (e.g. satellite-based, 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 Redirect, Predirect and unsolicited NA messages include only a single TLLAO with Link ID set to a constant value.

If the Client has multiple active underlying interfaces, then from the perspective of IPv6 ND it would appear to have a single link-local address with multiple link-layer addresses. In that case, Redirect, Predirect and unsolicited NA messages MAY include multiple TLLAOs -- each with a different Link ID that corresponds to a specific underlying interface of the Client.

3.5. AERO Interface Neighbor Cache Maintenace

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

Permanent neighbor cache entries are created through explicit administrative action; they have no timeout values and remain in place until explicitly deleted. AERO Relays maintain a permanent

neighbor cache entry for each Server on the link, and AERO Servers maintain a permanent neighbor cache entry for each Relay on the link.

Static neighbor cache entries are created though DHCPv6 PD exchanges and remain in place for durations bounded by prefix lifetimes. AERO Servers maintain a static neighbor cache entry for each of their associated Clients, and AERO Clients maintain a static neighbor cache for each of their associated Servers. When an AERO Server sends a DHCPv6 Reply message response to a Client's DHCPv6 Solicit or Renew message, it creates or updates a static neighbor cache entry based on the Client's AERO address as the network-layer address, the prefix lifetime as the neighbor cache entry lifetime, the Client's encapsulation IP address and UDP port number as the link-layer address and the prefix length as the length to apply to the AERO address. When an AERO Client receives a DHCPv6 Reply message from a Server, it creates or updates a static neighbor cache entry based on the Reply message link-local source address as the network-layer address, the prefix lifetime as the neighbor cache entry lifetime, and the encapsulation IP source address and UDP source port number as the link-layer address.

Dynamic neighbor cache entries are created based on receipt of an IPv6 ND message, and are garbage-collected if not used within a short timescale. AERO Clients maintain dynamic neighbor cache entries for each of their active correspondent Clients with lifetimes based on IPv6 ND messaging constants. When an AERO Client receives a valid Predirect message it creates or updates a dynamic neighbor cache entry for the Predirect target network-layer and link-layer addresses plus prefix length. The node then sets an "AcceptTime" variable for the neighbor and uses this value to determine whether packets received from the predirected neighbor can be accepted. When an AERO Client receives a valid Redirect message it creates or updates a dynamic neighbor cache entry for the Redirect target network-layer and link-layer addresses plus prefix length. The Client then sets a "ForwardTime" variable for the neighbor and uses this value to determine whether packets can be sent directly to the redirected neighbor. The Client also maintains a "MaxRetry" variable to limit the number of keepalives sent when a neighbor may have gone unreachable.

For dynamic neighbor cache entries, when an AERO Client receives a valid NS message it (re)sets AcceptTime for the neighbor to ACCEPT_TIME. When an AERO Client receives a valid solicited NA message, it (re)sets ForwardTime for the neighbor to FORWARD_TIME and sets MaxRetry to MAX_RETRY. When an AERO Client receives a valid unsolicited NA message, it updates the neighbor's link-layer addresses but DOES NOT reset AcceptTime, ForwardTime or MaxRetry.

It is RECOMMENDED that FORWARD_TIME be set to the default constant value 30 seconds to match the default REACHABLE_TIME value specified for IPv6 ND [RFC4861].

It is RECOMMENDED that ACCEPT_TIME be set to the default constant value 40 seconds to allow a 10 second window so that the AERO redirection procedure can converge before AcceptTime decrements below FORWARD_TIME.

It is RECOMMENDED that MAX_RETRY be set to 3 the same as described for IPv6 ND address resolution in Section 7.3.3 of [RFC4861].

Different values for FORWARD_TIME, ACCEPT_TIME, and MAX_RETRY MAY be administratively set, if necessary, to better match the AERO link's performance characteristics; however, if different values are chosen, all nodes on the link MUST consistently configure the same values. Most importantly, ACCEPT_TIME SHOULD be set to a value that is sufficiently longer than FORWARD_TIME to allow the AERO redirection procedure to converge.

3.6. AERO Interface Sending Algorithm

IP packets enter a node's AERO interface either from the network layer (i.e., from a local application or the IP forwarding system), or from the link-layer (i.e., from the AERO tunnel virtual link). Packets that enter the AERO interface from the network layer are encapsulated and admitted into the AERO link (i.e., they are tunnelled to an AERO interface neighbor). Packets that enter the AERO interface from the link layer are either re-admitted into the AERO link or delivered to the network layer where they are subject to either local delivery or IP forwarding. Since each AERO node has only partial information about neighbors on the link, AERO interfaces may forward packets with link-local destination addresses at a layer below the network layer. This means that AERO nodes act as both IP routers and link-layer "bridges". AERO interface sending considerations for Clients, Servers and Relays are given below.

When an IP packet enters a Client's AERO interface from the network layer, if the destination is covered by an ASP the Client searches for a dynamic neighbor cache entry with a non-zero ForwardTime and an AERO address that matches the packet's destination address. (The destination address may be either an address covered by the neighbor's ACP or the (link-local) AERO address itself.) If there is a match, the Client uses a link-layer address in the entry as the link-layer address for encapsulation then admits the packet into the AERO link. If there is no match, the Client instead uses the link-layer address of a neighboring Server as the link-layer address for encapsulation.

When an IP packet enters a Server's AERO interface from the link layer, if the destination is covered by an ASP the Server searches for a static neighbor cache entry with an AERO address that matches the packet's destination address. (The destination address may be either an address covered by the neighbor's ACP or the AERO address itself.) If there is a match, the Server uses a link-layer address in the entry as the link-layer address for encapsulation and re-admits the packet into the AERO link. If there is no match, the Server instead uses the link-layer address in any permanent neighbor cache entry as the link-layer address for encapsulation. When a Server receives a packet from a Relay, the Server MUST NOT loop the packet back to the same or a different Relay.

When an IP packet enters a Relay's AERO interface from the network layer, the Relay searches its IP forwarding table for an entry that is covered by an ASP and also matches the destination. If there is a match, the Relay uses the link-layer address in the neighbor cache entry for the next-hop Server as the link-layer address for encapsulation and admits the packet into the AERO link. When an IP packet enters a Relay's AERO interface from the link-layer, if the destination is not a link-local address and is not covered by an ASP the Relay removes the packet from the AERO interface and uses IP forwarding to forward the packet to the Internetwork. If the destination address is covered by an ASP, and there is a more-specific IP forwarding table entry that matches the destination, the Relay uses the link-layer address in the neighbor cache entry for the next-hop Server as the link-layer address for encapsulation and re-admits the packet into the AERO link. If there is no more-specific entry, the Relay instead drops the packet. When an Relay receives a packet from a Server, the Relay MUST NOT forward the packet back to the same Server.

Note that in the above that the link-layer address for encapsulation may be through consulting either the neighbor cache or the IP forwarding table. IP forwarding is therefore linked to IPv6 ND via the AERO address.

When an AERO node re-admits a packet into the AERO link, the node MUST NOT decrement the network layer TTL/Hop-count.

3.7. AERO Interface Encapsulation, Re-encapsulation and Decapsulation

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

AERO interfaces encapsulate packets per the specifications in [RFC2003][RFC2473][RFC4213][RFC4301][RFC5246] (etc.) except that the interface copies the "TTL/Hop Limit", "Type of Service/Traffic Class" and "Congestion Experienced" values in the packet's IP header into the corresponding fields in the encapsulation header. For packets undergoing re-encapsulation, the AERO interface instead copies the "TTL/Hop Limit", "Type of Service/Traffic Class" and "Congestion Experienced" values in the original encapsulation header into the corresponding fields in the new encapsulation header (i.e., the values are transferred between encapsulation headers and *not* copied from the encapsulated packet's network-layer header).

When AERO UDP encapsulation is used, the AERO interface encapsulates the packet per the above tunneling specifications except that it inserts a UDP header between the encapsulation header and the packet's IP header. The AERO interface sets the UDP source port to a constant value that it will use in each successive packet it sends, sets the UDP checksum field to zero (see: [RFC6935][RFC6936]) and sets the UDP length field to the length of the IP packet plus 8 bytes for the UDP header itself. For packets sent via a Server, the AERO interface sets the UDP destination port to 8060 (i.e., the IANA-registered port number for AERO) when AERO-only encapsulation is used. For packets sent to a neighboring Client, the AERO interface sets the UDP destination port to the port value stored in the neighbor cache entry for this neighbor.

The AERO interface next sets the IP protocol number in the encapsulation header to the appropriate value for the first protocol layer within the encapsulation (e.g., IPv4, IPv6, UDP, IPsec, etc.). When IPv6 is used as the encapsulation protocol, the interface then sets the flow label value in the encapsulation header the same as described in [RFC6438]. When IPv4 is used as the encapsulation protocol, the AERO interface sets the DF bit as discussed in Section 3.8.

AERO interfaces decapsulate packets destined either to the node itself or to a destination reached via an interface other than the AERO interface the packet was received on. When AERO UDP encapsulation is used (i.e., when a UDP header with destination port 8060 is present) the interface examines the first octet of the encapsulated packet. The packet is accepted if the most significant four bits of the first octet encode the value '0110' (i.e., the version number value for IPv6) or the value '0100' (i.e., the version number value for IPv4). Otherwise, the packet is accepted if the first octet encodes a valid IP protocol number per the IANA "protocol-numbers" registry that matches a supported encapsulation type. Otherwise, the packet is discarded.

Further decapsulation then proceeds according to the appropriate tunnel type per the above specifications.

3.8. AERO Interface Data Origin Authentication

AERO nodes employ simple data origin authentication procedures for encapsulated packets they receive from other nodes on the AERO link. In particular, AERO Clients accept encapsulated packets with a link-layer source address belonging to one of their current AERO Servers, and AERO Servers accept encapsulated packets with a link-layer source address belonging to one of their current Clients.

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

An AERO Server also accepts packets with a link-layer source address that matches one of its associated Relays, and an AERO Relay accepts packets with a source address that matches one of its associated Servers.

Finally, AERO Servers accept DHCPv6 messages even if the link-layer source address does not belong to one of their current Clients. The DHCPv6 server will authenticate the message and (assuming authentication succeeds) create or update a static neighbor cache entry for the source Client.

Note that this simple data origin authentication only applies to environments in which link-layer addresses cannot be spoofed. Additional security mitigations may be necessary in other environments.

3.9. AERO Interface MTU Considerations

The AERO link Maximum Transmission Unit (MTU) is 64KB minus the encapsulation overhead for IPv4 as the link-layer [RFC0791] and 4GB minus the encapsulation overhead for IPv6 as the link layer [RFC2675]. This is the most that IPv4 and IPv6 (respectively) can convey within the constraints of protocol constants, but actual sizes available for tunneling will frequently be much smaller.

The base tunneling specifications for IPv4 and IPv6 typically set a static MTU on the tunnel interface to 1500 bytes minus the encapsulation overhead or smaller still if the tunnel is likely to incur additional encapsulations on the path. This can result in path MTU related black holes when packets that are too large to be

accommodated over the AERO link are dropped, but the resulting ICMP Packet Too Big (PTB) messages are lost on the return path. As a result, AERO nodes use the following MTU mitigations to accommodate larger packets.

AERO nodes set their AERO interface MTU to the larger of the underlying interface MTU minus the encapsulation overhead, and 1500 bytes. (If there are multiple underlying interfaces, the node sets the AERO interface MTU according to the largest underlying interface MTU, or 64KB /4G minus the encapsulation overhead if the largest MTU cannot be determined.) AERO nodes optionally cache other per-neighbor MTU values in the underlying IP path MTU discovery cache initialized to the underlying interface MTU.

AERO nodes admit packets that are no larger than 1280 bytes minus the encapsulation overhead (*) as well as packets that are larger than 1500 bytes into the tunnel without fragmentation, i.e., as long as they are no larger than the AERO interface MTU before encapsulation and also no larger than the cached per-neighbor MTU following encapsulation. For IPv4, the node sets the "Don't Fragment" (DF) bit to 0 for packets no larger than 1280 bytes minus the encapsulation overhead (*) and sets the DF bit to 1 for packets larger than 1500 bytes. If a large packet is lost in the path, the node may optionally cache the MTU reported in the resulting PTB message or may ignore the message, e.g., if there is a possibility that the message is spurious.

For packets destined to an AERO node that are larger than 1280 bytes minus the encapsulation overhead (*) but no larger than 1500 bytes, the node uses IP fragmentation to fragment the encapsulated packet into two pieces (where the first fragment contains 1024 bytes of the original IP packet) then admits the fragments into the tunnel. If the link-layer protocol is IPv4, the node admits each fragment into the tunnel with DF set to 0 and subject to rate limiting to avoid reassembly errors [RFC4963][RFC6864]. For both IPv4 and IPv6, the node also sends a 1500 byte probe message (**) to the neighbor, subject to rate limiting.

To construct a probe, the node prepares an NS message with a Nonce option plus trailing padding octets added to a length of 1500 bytes without including the length of the padding in the IPv6 Payload Length field. The node then encapsulates the NS in the encapsulation headers (while including the length of the padding in the encapsulation header length fields), sets DF to 1 (for IPv4) and sends the padded NS message to the neighbor. If the neighbor returns an NA message with a correct Nonce value, the node may then send whole packets within this size range and (for IPv4) relax the rate limiting requirement. (Note that the trailing padding SHOULD NOT be

included within the Nonce option itself but rather as padding beyond the last option in the NS message; otherwise, the (large) Nonce option would be echoed back in the solicited NA message and may be lost at a link with a small MTU along the reverse path.)

AERO nodes MUST be capable of reassembling packets up to 1500 bytes plus the encapsulation overhead length. It is therefore RECOMMENDED that AERO nodes be capable of reassembling at least 2KB.

(*) Note that if it is known without probing that the minimum Path MTU to an AERO node is MINMTU bytes (where 1280 < MINMTU < 1500) then MINMTU can be used instead of 1280 in the fragmentation threshold considerations listed above.

(**) It is RECOMMENDED that no probes smaller than 1500 bytes be used for MTU probing purposes, since smaller probes may be fragmented if there is a nested tunnel somewhere on the path to the neighbor. Probe sizes larger than 1500 bytes MAY be used, but may be unnecessary since original sources are expected to implement [RFC4821] when sending large packets.

3.10. AERO Router Discovery, Prefix Delegation and Address Configuration

3.10.1. AERO DHCPv6 Service Model

Each AERO Server configures a DHCPv6 server function to facilitate PD requests from Clients. Each Server is pre-configured with an identical list of ACP-to-Client ID mappings for all Clients enrolled in the AERO system, as well as any information necessary to authenticate Clients. The configuration information is maintained by a central administrative authority for the AERO link and securely propagated to all Servers whenever a new Client is enrolled or an existing Client is withdrawn.

With these identical configurations, each Server can function independently of all other Servers, including the maintenance of active leases. Therefore, no Server-to-Server DHCPv6 state synchronization is necessary, and Clients can optionally hold separate leases for the same ACP from multiple Servers.

In this way, Clients can easily associate with multiple Servers, and can receive new leases from new Servers before deprecating leases held through old Servers. This enables a graceful "make-before-break" capability.

3.10.2. AERO Client Behavior

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

To associate with a Server, the Client acts as a requesting router to request an ACP through a DHCPv6 PD two-message exchange[RFC3315][RFC3633] in which the Solicit message uses the IPv6 "unspecified" address (i.e., "::") as the IPv6 source address, 'All_DHCP_Relay_Agents_and_Servers' as the IPv6 destination address and the link-layer address of the Server as the link-layer destination address. The Client includes a Rapid Commit option as well as a Client Identifier option with a DHCP Unique Identifier (DUID), plus any necessary authentication options to identify itself to the DHCPv6 server. The Client also includes a Client Link Layer Address Option (CLLAO) [RFC6939] with the format shown in Figure 3

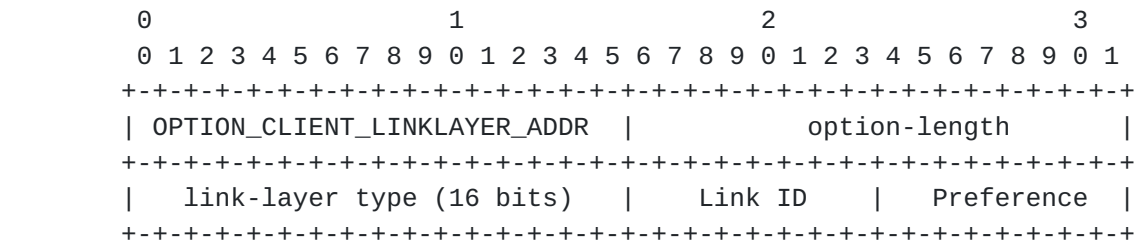

Figure 3: AERO Client Link-Layer Address Option (CLLAO) Format

The Client sets the CLLAO 'option-length' field to 4 and sets the 'link-layer type' field to TBD1 (see: IANA Considerations), then includes appropriate Link ID and Preference values for the underlying interface over which the Solicit will be issued (note that these are the same values that would be included in a TLLAO as shown in Figure 2). If the Client is pre-provisioned with an ACP associated with the AERO service, it MAY also include the ACP in the Solicit message Identity Association (IA) option to indicate its preferred ACP to the DHCPv6 server. The Client then sends the encapsulated DHCPv6 request via the underlying interface.

When the Client receives its ACP and the set of ASPs via a DHCPv6 Reply from the AERO Server, it creates a static neighbor cache entry with the Server's link-local address (i.e., fe80::ID) as the network-

layer address and the Server's encapsulation address as the link-layer address. The Client then records the lifetime for the ACP in the neighbor cache entry and marks the neighbor cache entry as "default", i.e., the Client considers the Server as a default router. If the Reply message contains a Vendor-Specific Information Option (see: Section 3.10.3) the Client also caches each ASP in the option.

The Client then applies the AERO address to the AERO interface and sub-delegates the ACP to nodes and links within its attached EUNs (the AERO address thereafter remains stable as the Client moves). The Client also assigns a default IP route to the AERO interface as a route-to-interface, i.e., with no explicit next-hop. The next hop will then be determined after a packet has been submitted to the AERO interface by inspecting the neighbor cache (see above).

The Client subsequently renews its ACP delegation through each of its Servers by performing DHCPv6 Renew/Reply exchanges with its AERO address as the IPv6 source address, 'All_DHCP_Relay_Agents_and_Servers' as the IPv6 destination address, the link-layer address of a Server as the link-layer destination address and the same Client identifier, authentication options and CLLAO option as was used in the initial PD request.

Since the Client's AERO address is configured from the unique ACP delegation it receives, there is no need for Duplicate Address Detection (DAD) on AERO links. Other nodes maliciously attempting to hijack an authorized Client's AERO address will be denied access to the network by the DHCPv6 server due to an unacceptable link-layer address and/or security parameters (see: Security Considerations).

AERO Clients ignore the IP address and UDP port number in any S/TLLAO options in ND messages they receive directly from another AERO Client, but examine the Link ID and Preference values to match the message with the correct link-layer address information.

When a source Client forwards a packet to a prospective destination Client (i.e., one for which the packet's destination address is covered by an ASP), the source Client initiates an AERO route optimization procedure as specified in Section 3.12.

3.10.3. AERO Server Behavior

AERO Servers configure a DHCPv6 server function on their AERO links. AERO Servers arrange to add their encapsulation layer IP addresses (i.e., their link-layer addresses) to the DNS resource records for the FQDN "linkupnetworks.[domainname]" before entering service.

When an AERO Server receives a prospective Client's DHCPv6 PD Solicit message, it first authenticates the message. If authentication succeeds, the Server determines the correct ACP to delegate to the Client by matching the Client's DUID within an online directory service (e.g., LDAP). The Server then delegates the ACP and creates a static neighbor cache entry for the Client's AERO address with lifetime set to no more than the lease lifetime and the Client's link-layer address as the link-layer address for the Link ID specified in the CLLAO option. The Server then creates an IP forwarding table entry so that the inter-Server/Relay routing system will propagate the ACP to all Relays (see: Section 3.11). Finally, the Server sends a DHCPv6 Reply message to the Client while using fe80::ID as the IPv6 source address, the Client's AERO address as the IPv6 destination address, and the Client's link-layer address as the destination link-layer address. The Server also includes a Server Unicast option with server-address set to fe80::ID so that all future Client/Server transactions will be link-local-only unicast over the AERO link.

When the Server sends the DHCPv6 Reply message, it also includes a DHCPv6 Vendor-Specific Information Option with 'enterprise-number' set to "TBD2" (see: IANA Considerations). The option is formatted as shown in[RFC3315] and with the AERO enterprise-specific format shown in Figure 4:

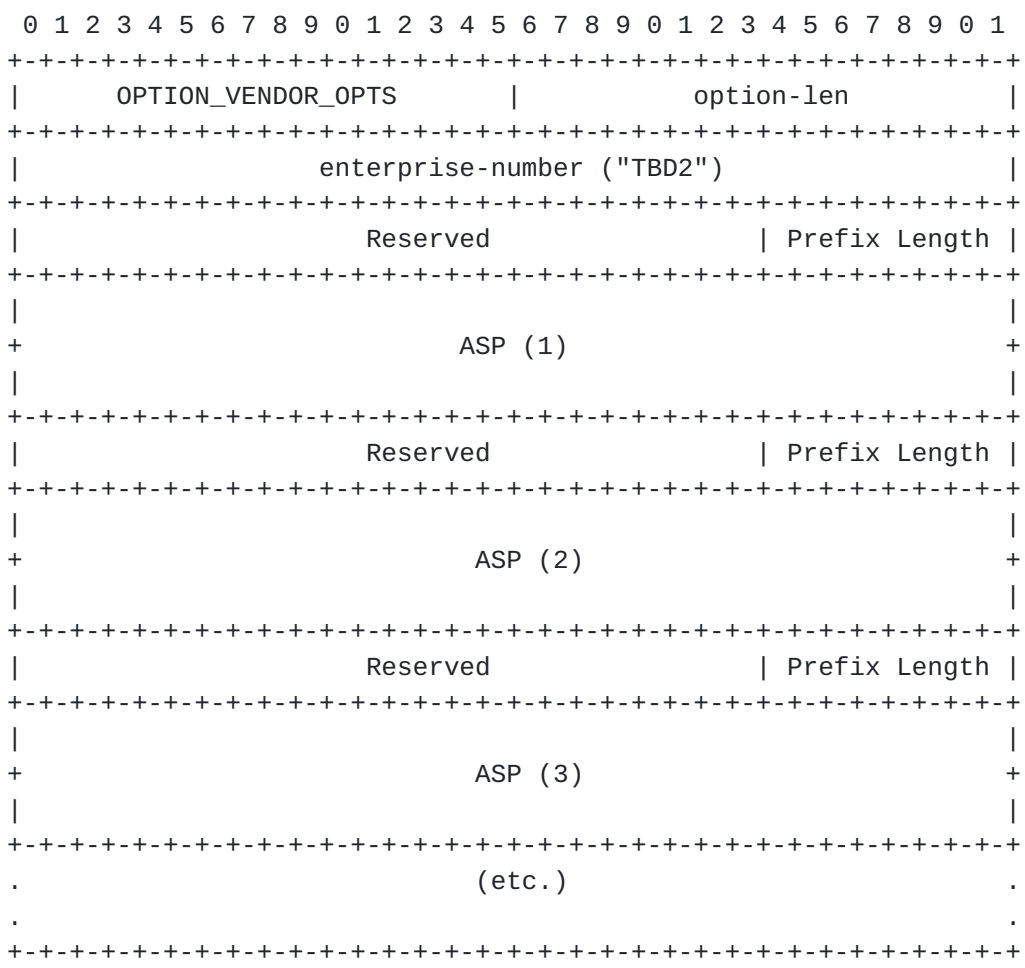

Figure 4: AERO Vendor-Specific Information Option

Per Figure 4, the option includes one or more ASP. The ASP field contains the IP prefix as it would appear in the interface identifier portion of the corresponding AERO address (see: Section 3.3). For IPv6, valid values for the Prefix Length field are 0 through 64; for IPv4, valid values are 0 through 32.

After the initial Solicit/Reply exchange, the AERO Server maintains the neighbor cache entry for the Client as long as the lease lifetime remains current. If the Client issues a Renew/Reply exchange, the Server extends the lifetime. If the Client issues a Release/Reply exchange, or if the Client does not issue a Renew/Reply within the lease lifetime, the Server deletes the neighbor cache entry for the Client and withdraws the IP route from the routing system.

3.11. AERO Relay/Server Routing System

Relays require full topology information of all Client/Server associations, while individual Servers only require partial topology information, i.e., they only need to know the ACPs associated with their current set of associated Clients. This is accomplished through the use of an internal instance of the Border Gateway Protocol (BGP) [RFC4271] coordinated between Servers and Relays. This internal BGP instance does not interact with the public Internet BGP instance; therefore, the AERO link is presented to the IP Internetwork as a small set of ASPs as opposed to the full set of individual ACPs.

In a reference BGP arrangement, each AERO Server is configured as an Autonomous System Border Router (ASBR) for a stub Autonomous System (AS) (possibly using a private AS Number (ASN) [RFC1930]), and each Server further peers with each Relay but does not peer with other Servers. Similarly, Relays need not peer with each other, since they will receive all updates from all Servers and will therefore have a consistent view of the AERO link ACP delegations.

Each Server maintains a working set of associated Clients, and dynamically announces new ACPs and withdraws departed ACPs in its BGP updates to Relays. Relays do not send BGP updates to Servers, however, such that the BGP route reporting is unidirectional from the Servers to the Relays.

The Relays therefore discover the full topology of the AERO link in terms of the working set of ACPs associated with each Server, while the Servers only discover the ACPs of their associated Clients. Since Clients are expected to remain associated with their current set of Servers for extended timeframes, the amount of BGP control messaging between Servers and Relays should be minimal. However, BGP peers SHOULD dampen any route oscillations caused by impatient Clients that repeatedly associate and disassociate with Servers.

3.12. AERO Redirection

3.12.1. Reference Operational Scenario

Figure 5 depicts the AERO redirection reference operational scenario, using IPv6 addressing as the example (while not shown, a corresponding example for IPv4 addressing can be easily constructed). The figure shows an AERO Relay ('R'), two AERO Servers ('S1', 'S2'), two AERO Clients ('A', 'B') and two ordinary IPv6 hosts ('C', 'D'):

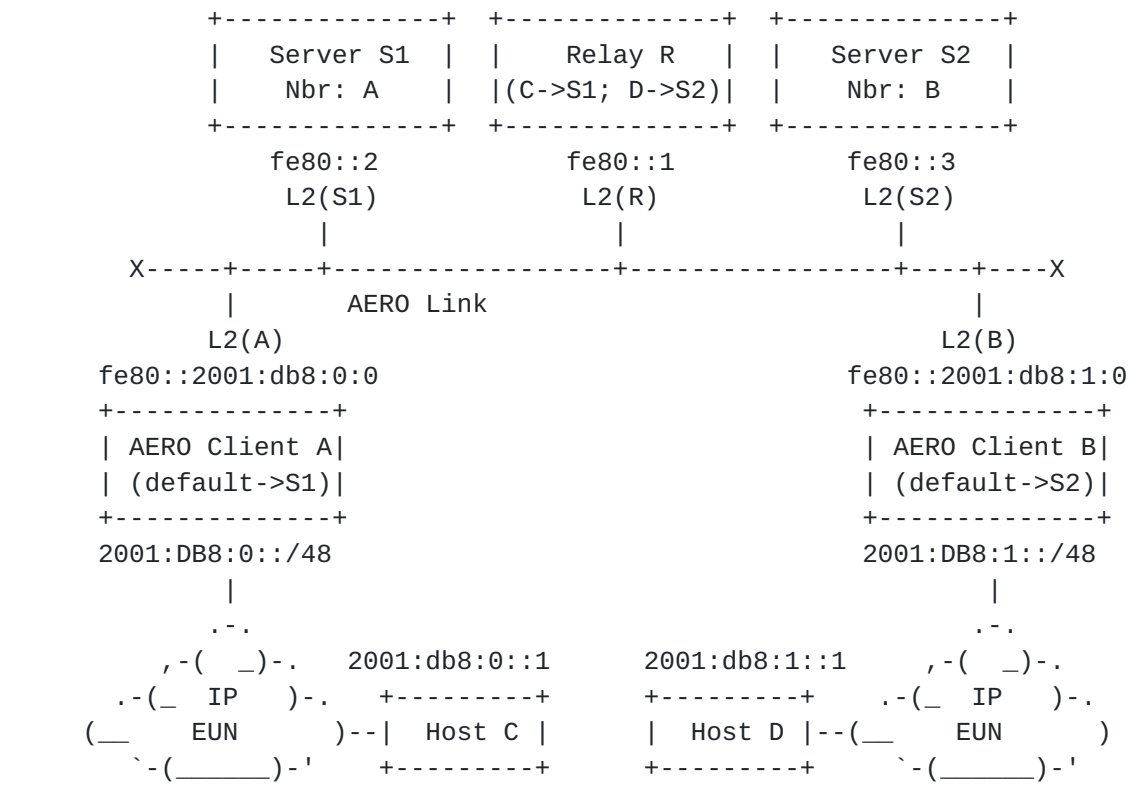

Figure 5: AERO Reference Operational Scenario

In Figure 5, Relay ('R') applies the address fe80::1 to its AERO interface with link-layer address L2(R), Server ('S1') applies the address fe80::2 with link-layer address L2(S1),and Server ('S2') applies the address fe80::3 with link-layer address L2(S2). Servers ('S1') and ('S2') next arrange to add their link-layer addresses to a published list of valid Servers for the AERO link.

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

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

EUNs. IPv6 host ('D') connects to the EUN, and configures the address 2001:db8:1::1.

3.12.2. Concept of Operations

Again, with reference to Figure 5, when source host ('C') sends a packet to destination host ('D'), the packet is first forwarded over the source host's attached EUN to Client ('A'). Client ('A') then forwards the packet via its AERO interface to Server ('S1') and also sends a Predirect message toward Client ('B') via Server ('S1'). Server ('S1') then re-encapsulates and forwards both the packet and the Predirect message out the same AERO interface toward Client ('B') via Relay ('R').

When Relay ('R') receives the packet and Predirect message, it consults its forwarding table to discover Server ('S2') as the next hop toward Client ('B'). Relay ('R') then forwards both the packet and the Predirect message to Server ('S2'), which then forwards them to Client ('B').

After Client ('B') receives the Predirect message, it process the message and returns a Redirect message toward Client ('A') via Server ('S2'). During the process, Client ('B') also creates or updates a dynamic neighbor cache entry for Client ('A').

When Server ('S2') receives the Redirect message, it re-encapsulates the message and forwards it on to Relay ('R'), which forwards the message on to Server ('S1') which forwards the message on to Client ('A'). After Client ('A') receives the Redirect message, it processes the message and creates or updates a dynamic neighbor cache entry for Client ('C').

Following the above Predirect/Redirect message exchange, forwarding of packets from Client ('A') to Client ('B') without involving any intermediate nodes is enabled. The mechanisms that support this exchange are specified in the following sections.

3.12.3. Message Format

AERO Redirect/Predirect messages use the same format as for ICMPv6 Redirect messages depicted in Section 4.5 of [RFC4861], but also include a new "Prefix Length" field taken from the low-order 8 bits of the Redirect message Reserved field. For IPv6, valid values for the Prefix Length field are 0 through 64; for IPv4, valid values are 0 through 32. The Redirect/Predirect messages are formatted as shown in Figure 6:

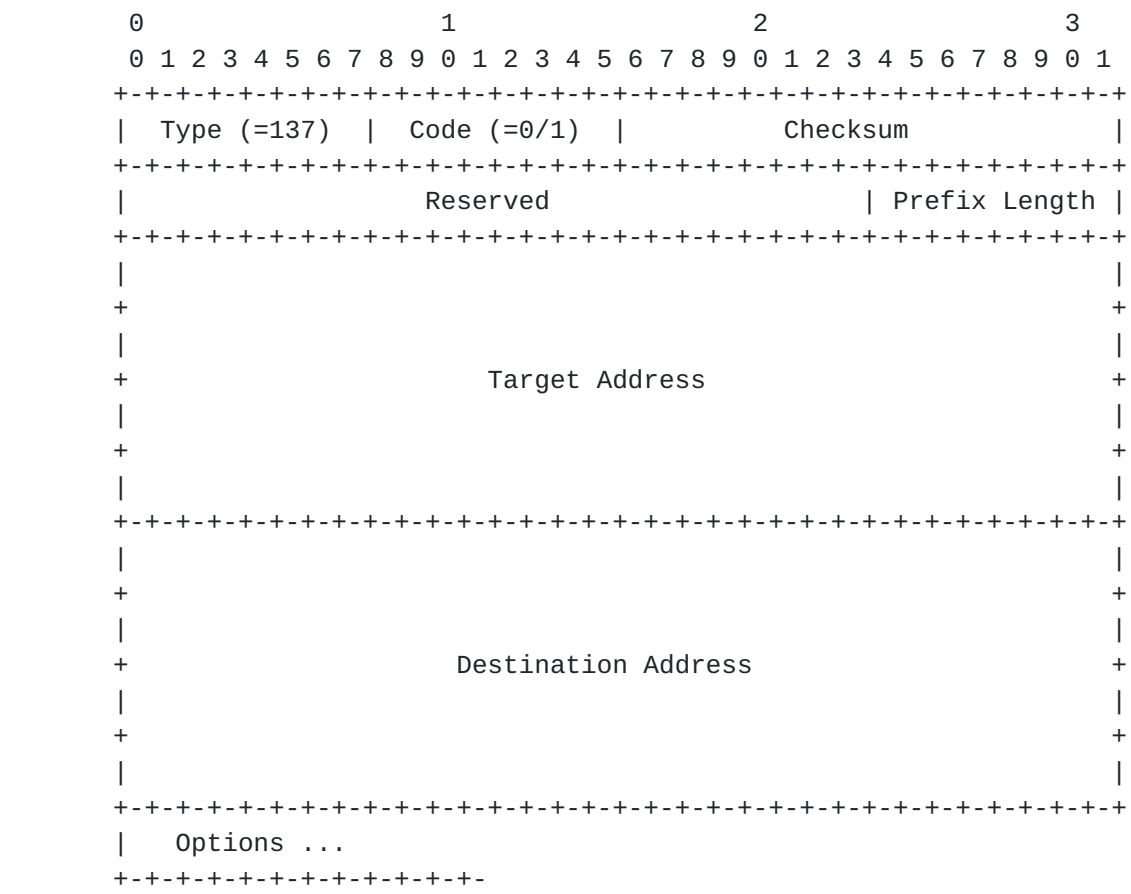

Figure 6: AERO Redirect/Predirect Message Format

3.12.4. Sending Predirects

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

In the reference operational scenario, when Client ('A') forwards a packet toward Client ('B'), it MAY also send a Predirect message forward toward Client ('B'), subject to rate limiting (see Section 8.2 of [RFC4861]). Client ('A') prepares the Predirect message as follows:

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

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

- the network-layer source address is set to fe80::2001:db8:0:0 (i.e., the AERO address of Client ('A')).
- the network-layer destination address is set to fe80::2001:db8:1:0 (i.e., the AERO address of Client ('B')).
- the Type is set to 137.
- the Code is set to 1 to indicate "Predirect".
- the Prefix Length is set to the length of the prefix to be applied to the Target Address.
- the Target Address is set to fe80::2001:db8:0:0 (i.e., the AERO address of Client ('A')).
- the Destination Address is set to the source address of the originating packet that triggered the Predirection event. (If the originating packet is an IPv4 packet, the address is constructed in IPv4-compatible IPv6 address format).
- the message includes one or more TLLAOs with Link ID and Preference set to appropriate values for Client ('A')'s underlying interfaces, and with UDP Port Number and IP Address set to 0'.
- the message SHOULD include a Timestamp option and a Nonce option.
- the message includes a Redirected Header Option (RHO) that contains the originating packet truncated to ensure that at least the network-layer header is included but the size of the message does not exceed 1280 bytes.

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

3.12.5. Re-encapsulating and Relaying Predirects

When Server ('S1') receives a Predirect message from Client ('A'), it first verifies that the TLLAOs in the Predirect are a proper subset of the Link IDs in Client ('A')'s neighbor cache entry. If the Client's TLLAOs are not acceptable, Server ('S1') discards the message. Otherwise, Server ('S1') validates the message according to the ICMPv6 Redirect message validation rules in Section 8.1 of [RFC4861], except that the Predirect has Code=1. Server ('S1') also

verifies that Client ('A') is authorized to use the Prefix Length in the Predirect when applied to the AERO address in the network-layer source address by searching for the AERO address in the neighbor cache. If validation fails, Server ('S1') discards the Predirect; otherwise, it copies the correct UDP Port numbers and IP Addresses for Client ('A')'s links into the (previously empty) TLLAOs.

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

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

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

3.12.6. Processing Predirects and Sending Redirects

When Client ('B') receives the Predirect message, it accepts the Predirect only if the message has a link-layer source address of one of its Servers (e.g., L2(S2)). Client ('B') further accepts the message only if it is willing to serve as a redirection target. Next, Client ('B') validates the message according to the ICMPv6 Redirect message validation rules in Section 8.1 of [RFC4861], except that it accepts the message even though Code=1 and even though the network-layer source address is not that of it's current first-hop router.

In the reference operational scenario, when Client ('B') receives a valid Predirect message, it either creates or updates a dynamic neighbor cache entry that stores the Target Address of the message as the network-layer address of Client ('A') , stores the link-layer addresses found in the TLLAOs as the link-layer addresses of Client

('A') and stores the Prefix Length as the length to be applied to the network-layer address for forwarding purposes. Client ('B') then sets AcceptTime for the neighbor cache entry to ACCEPT_TIME.

After processing the message, Client ('B') prepares a Redirect message response as follows:

- the link-layer source address is set to 'L2(B)' (i.e., the link-layer address of Client ('B')).
- the link-layer destination address is set to 'L2(S2)' (i.e., the link-layer address of Server ('S2')).
- the network-layer source address is set to fe80::2001:db8:1:0 (i.e., the AERO address of Client ('B')).
- the network-layer destination address is set to fe80::2001:db8:0:0 (i.e., the AERO address of Client ('A')).
- the Type is set to 137.
- the Code is set to 0 to indicate "Redirect".
- the Prefix Length is set to the length of the prefix to be applied to the Target Address.
- the Target Address is set to fe80::2001:db8:1:0 (i.e., the AERO address of Client ('B')).
- the Destination Address is set to the destination address of the originating packet that triggered the Redirection event. (If the originating packet is an IPv4 packet, the address is constructed in IPv4-compatible IPv6 address format).
- the message includes one or more TLLAOs with Link ID and Preference set to appropriate values for Client ('B')'s underlying interfaces, and with UDP Port Number and IP Address set to '0'.
- the message SHOULD include a Timestamp option and MUST echo the Nonce option received in the Predirect (i.e., if a Nonce option is included).
- the message includes as much of the RHO copied from the corresponding AERO Predirect message as possible such that at least the network-layer header is included but the size of the message does not exceed 1280 bytes.

After Client ('B') prepares the Redirect message, it sends the message to Server ('S2').

3.12.7. Re-encapsulating and Relaying Redirects

When Server ('S2') receives a Redirect message from Client ('B'), it first verifies that the TLLAOs in the Redirect are a proper subset of the Link IDs in Client ('B')'s neighbor cache entry. If the Client's TLLAOs are not acceptable, Server ('S2') discards the message. Otherwise, Server ('S2') validates the message according to the ICMPv6 Redirect message validation rules in Section 8.1 of [RFC4861]. Server ('S2') also verifies that Client ('B') is authorized to use the Prefix Length in the Redirect when applied to the AERO address in the network-layer source address by searching for the AERO address in the neighbor cache. If validation fails, Server ('S2') discards the Predirect; otherwise, it copies the correct UDP Port numbers and IP Addresses for Client ('B')'s links into the (previously empty) TLLAOs.

Server ('S2') then examines the network-layer destination address of the Predirect to determine the next hop toward Client ('A') by searching for the AERO address in the neighbor cache. Since Client ('A') is not one of its neighbors, Server ('S2') re-encapsulates the Predirect and relays it via Relay ('R') by changing the link-layer source address of the message to 'L2(S2)' and changing the link-layer destination address to 'L2(R)'. Server ('S2') finally forwards the re-encapsulated message to Relay ('R') without decrementing the network-layer TTL/Hop Limit field.

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

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

3.12.8. Processing Redirects

When Client ('A') receives the Redirect message, it accepts the message only if it has a link-layer source address of one of its Servers (e.g., ''L2(S1)'). Next, Client ('A') validates the message

according to the ICMPv6 Redirect message validation rules in Section 8.1 of [RFC4861], except that it accepts the message even though the network-layer source address is not that of it's current first-hop router. Following validation, Client ('A') then processes the message as follows.

In the reference operational scenario, when Client ('A') receives the Redirect message, it either creates or updates a dynamic neighbor cache entry that stores the Target Address of the message as the network-layer address of Client ('B'), stores the link-layer addresses found in the TLLAOs as the link-layer addresses of Client ('B') and stores the Prefix Length as the length to be applied to the network-layer address for forwarding purposes. Client ('A') then sets ForwardTime for the neighbor cache entry to FORWARD_TIME.

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

3.12.9. Server-Oriented Redirection

In some environments, the Server nearest the target Client may need to serve as the redirection target, e.g., if direct Client-to-Client communications are not possible. In that case, the Server prepares the Redirect message the same as if it were the destination Client (see: Section 3.9.6), except that it writes its own link-layer address in the TLLAO option. The Server must then maintain a neighbor cache entry for the redirected source Client.

3.13. Neighbor Unreachability Detection (NUD)

AERO nodes perform NUD by sending unicast NS messages to elicit solicited NA messages from neighbors the same as described in [RFC4861]. When an AERO node sends an NS/NA message, it MUST use its link-local address as the IPv6 source address and the link-local address of the neighbor as the IPv6 destination address. When an AERO node receives an NS message or a solicited NA message, it accepts the message if it has a neighbor cache entry for the neighbor; otherwise, it ignores the message.

When a source Client is redirected to a target Client it SHOULD test the direct path by sending an initial NS message to elicit a

solicited NA response. While testing the path, the source Client can optionally continue sending packets via the Server, maintain a small queue of packets until target reachability is confirmed, or (optimistically) allow packets to flow directly to the target. The source Client SHOULD thereafter continue to test the direct path to the target Client (see Section 7.3 of [RFC4861]) periodically in order to keep dynamic neighbor cache entries alive.

In particular, while the source Client is actively sending packets to the target Client it SHOULD also send NS messages separated by RETRANS_TIMER milliseconds in order to receive solicited NA messages. If the source Client is unable to elicit a solicited NA response from the target Client after MAX_RETRY attempts, it SHOULD set ForwardTime to 0 and resume sending packets via one of its Servers. Otherwise, the source Client considers the path usable and SHOULD thereafter process any link-layer errors as a hint that the direct path to the target Client has either failed or has become intermittent.

When a target Client receives an NS message from a source Client, it resets AcceptTime to ACCEPT_TIME if a neighbor cache entry exists; otherwise, it discards the NS message. If ForwardTime is non-zero, the target Client then sends a solicited NA message to the link-layer address of the source Client; otherwise, it sends the solicited NA message to the link-layer address of one of its Servers.

When a source Client receives a solicited NA message from a target Client, it resets ForwardTime to FORWARD_TIME if a neighbor cache entry exists; otherwise, it discards the NA message.

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

3.14. Mobility Management

3.14.1. Announcing Link-Layer Address Changes

When a Client needs to change its link-layer address, e.g., due to a mobility event, it performs an immediate DHCPv6 Rebind/Reply exchange via each of its Servers using the new link-layer address as the source and with a CLLAO that includes the correct Link ID and Preference values. If authentication succeeds, the Server then update its neighbor cache and sends a DHCPv6 Reply.

Next, the Client sends unsolicited NA messages to each of its correspondent Client neighbors using the same procedures as specified in Section 7.2.6 of [RFC4861], except that it sends the messages as unicast to each neighbor via a Server instead of multicast. In this process, the Client should send no more than MAX_NEIGHBOR_ADVERTISEMENT messages separated by no less than RETRANS_TIMER seconds to each neighbor.

With reference to Figure 5, Client ('B') sends unicast unsolicited NA messages to Client ('A') via Server ('S2') as follows:

- the link-layer source address is set to 'L2(B)' (i.e., the link-layer address of Client ('B')).
- the link-layer destination address is set to 'L2(S2)' (i.e., the link-layer address of Server ('S2')).
- the network-layer source address is set to fe80::2001:db8:1:0 (i.e., the AERO address of Client ('B')).
- the network-layer destination address is set to fe80::2001:db8:0:0 (i.e., the AERO address of Client ('A')).
- the Type is set to 136.
- the Code is set to 0.
- the Solicited flag is set to 0.
- the Override flag is set to 1.
- the Target Address is set to fe80::2001:db8:1:0 (i.e., the AERO address of Client ('B')).
- the message includes one or more TLLAOs with Link ID and Preference set to appropriate values for Client ('B')'s underlying interfaces, and with UDP Port Number and IP Address set to '0'.
- the message SHOULD include a Timestamp option.

When Server ('S1') receives the NA message, it relays the message in the same way as described for relaying Redirect messages in Section 3.12.7. In particular, Server ('S1') copies the correct UDP port numbers and IP addresses into the TLLAOs, changes the link-layer source address to its own address, changes the link-layer destination address to the address of Relay ('R'), then forwards the NA message via the relaying chain the same as for a Redirect.

When Client ('A') receives the NA message, it accepts the message only if it already has a neighbor cache entry for Client ('B') then updates the link-layer addresses for Client ('B') based on the addresses in the TLLAOs. However, Client ('A') MUST NOT update ForwardTime since Client ('B') will not have updated AcceptTime.

Note that these unsolicited NA messages are unacknowledged; hence, Client ('B') has no way of knowing whether Client ('A') has received them. If the messages are somehow lost, however, Client ('A') will soon learn of the mobility event via the NUD procedures specified in Section 3.13.

3.14.2. Bringing New Links Into Service

When a Client needs to bring a new underlying interface into service (e.g., when it activates a new data link), it performs an immediate Rebind/Reply exchange via each of its Servers using the new link-layer address as the source address and with a CLLAO that includes the new Link ID and Preference values. If authentication succeeds, the Server then updates its neighbor cache and sends a DHCPv6 Reply. The Client MAY then send unsolicited NA messages to each of its correspondent Clients to inform them of the new link-layer address as described in Section 3.14.1.

3.14.3. Removing Existing Links from Service

When a Client needs to remove an existing underlying interface from service (e.g., when it de-activates an existing data link), it performs an immediate Rebind/Reply exchange via each of its Servers over any available link with a CLLAO that includes the deprecated Link ID and a Preference value of 0. If authentication succeeds, the Server then updates its neighbor cache and sends a DHCPv6 Reply. The Client SHOULD then send unsolicited NA messages to each of its correspondent Clients to inform them of the deprecated link-layer address as described in Section 3.14.1.

3.14.4. Moving to a New Server

When a Client associates with a new Server, it performs the Client procedures specified in Section 3.10.

When a Client disassociates with an existing Server, it sends a DHCPv6 Release message to the unicast network layer address of the old Server. The Client SHOULD send the message via a new Server (i.e., by setting the link-layer destination address to the address of the new Server) in case the old Server is unreachable at the link layer, e.g., if the old Server is in a different network partition.

The new Server will forward the message to a Relay, which will in turn forward the message to the old Server.

When the old Server receives the DHCPv6 Release, it first authenticates the message. If authentication succeeds, the old Server withdraws the IP route from the routing system and deletes the neighbor cache entry for the Client. (The old Server MAY impose a small delay before deleting the neighbor cache entry so that any packets already in the system can still be delivered to the Client.) The old Server then returns a DHCPv6 Reply message via a Relay. The Client can then use the Reply message to verify that the termination signal has been processed, and can delete both the default route and the neighbor cache entry for the old Server.

Clients SHOULD NOT move rapidly between Servers in order to avoid causing unpredictable oscillations in the Server/Relay routing system. Such oscillations could result in intermittent reachability for the Client itself, while causing little harm to the network due to routing protocol dampening. Examples of when a Client might wish to change to a different Server include a Server that has gone unreachable, topological movements of significant distance, etc.

3.15. Encapsulation Protocol Version Considerations

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

3.16. Multicast Considerations

When the underlying network does not support multicast, AERO nodes map IPv6 link-scoped multicast addresses (including 'All_DHCP_Relay_Agents_and_Servers') to the link-layer address of a Server.

When the underlying network supports multicast, AERO nodes use the multicast address mapping specification found in [RFC2529] for IPv4 underlying networks and use a direct multicast mapping for IPv6 underlying networks. (In the latter case, "direct multicast mapping" means that if the IPv6 multicast destination address of the encapsulated packet is "M", then the IPv6 multicast destination address of the encapsulating header is also "M".)

3.17. Operation on AERO Links Without DHCPv6 Services

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

3.18. Operation on Server-less AERO Links

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

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

4. Implementation Status

An application-layer implementation is in progress.

5. IANA Considerations

IANA is instructed to assign a new 2-octet Hardware Type number "TBD1" for AERO in the "arp-parameters" registry per Section 2 of [RFC5494]. The number is assigned from the 2-octet Unassigned range with Hardware Type "AERO" and with this document as the reference.

IANA is instructed to assign a 4-octet Enterprise Number "TBD2" for AERO in the "enterprise-numbers" registry per [RFC3315].

6. Security Considerations

AERO link security considerations are the same as for standard IPv6 Neighbor Discovery [RFC4861] except that AERO improves on some aspects. In particular, AERO uses a trust basis between Clients and Servers, where the Clients only engage in the AERO mechanism when it is facilitated by a trust anchor. AERO nodes SHOULD also use DHCPv6 securing services (e.g., DHCPv6 authentication,

[I-D.ietf-dhc-sedhcpv6], etc.) for Client authentication and network admission control.

AERO Redirect, Predirect and unsolicited NA messages SHOULD include a Timestamp option (see Section 5.3 of [RFC3971]) that other AERO nodes can use to verify the message time of origin. AERO Predirect, NS and RS messages SHOULD include a Nonce option (see Section 5.3 of [RFC3971]) that recipients echo back in corresponding responses.

AERO links must be protected against link-layer address spoofing attacks in which an attacker on the link pretends to be a trusted neighbor. Links that provide link-layer securing mechanisms (e.g., IEEE 802.1X WLANs) and links that provide physical security (e.g., enterprise network wired LANs) provide a first line of defense that is often sufficient. In other instances, additional securing mechanisms such as Secure Neighbor Discovery (SeND) [RFC3971], IPsec [RFC4301] or TLS [RFC5246] may be necessary.

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 unauthorized nodes via a NAT function.)

On some AERO links, establishment and maintenance of a direct path between neighbors requires secured coordination such as through the Internet Key Exchange (IKEv2) protocol [RFC5996] to establish a security association.

7. Acknowledgements

Discussions both on IETF lists and in private exchanges helped shape some of the concepts in this work. Individuals who contributed insights include Mikael Abrahamsson, Fred Baker, Stewart Bryant, Brian Carpenter, Wojciech Dec, Ralph Droms, Brian Haberman, Joel Halpern, Sascha Hlusiak, Lee Howard, Andre Kostur, Ted Lemon, Joe Touch and Bernie Volz. 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.

This work has further been encouraged and supported by Boeing colleagues including Keith Bartley, Dave Bernhardt, Cam Brodie, Balaguruna Chidambaram, Claudiu Danilov, Wen Fang, Anthony Gregory, Jeff Holland, Ed King, Gen MacLean, Kent Shuey, Brian Skeen, Mike

Slane, Julie Wulff, Yueli Yang, and other members of the BR&T and BIT mobile networking teams.

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

8. References

8.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC6434] Jankiewicz, E., Loughney, J., and T. Narten, "IPv6 Node Requirements", RFC 6434, December 2011.

8.2. Informative References

[I-D.ietf-dhc-sedhcpv6] Jiang, S., Shen, S., Zhang, D., and T. Jinmei, "Secure DHCPv6 with Public Key", draft-ietf-dhc-sedhcpv6-03 (work in progress), June 2014.

[RFC0879] Postel, J., "TCP maximum segment size and related topics", RFC 879, November 1983.

[RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", BCP 6, RFC 1930, March 1996.

[RFC2529] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529, March 1999.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, August 1999.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.

[RFC4994] Zeng, S., Volz, B., Kinnear, K., and J. Brzozowski, "DHCPv6 Relay Agent Echo Request Option", RFC 4994, September 2007.

[RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, March 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5494] Arkko, J. and C. Pignataro, "IANA Allocation Guidelines for the Address Resolution Protocol (ARP)", RFC 5494, April 2009.

[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, October 2009.

[RFC5569] Despres, R., "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)", RFC 5569, January 2010.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6204] Singh, H., Beebee, W., Donley, C., Stark, B., and O. Troan, "Basic Requirements for IPv6 Customer Edge Routers", RFC 6204, April 2011.

[RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, August 2011.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, November 2011.

[RFC6691] Borman, D., "TCP Options and Maximum Segment Size (MSS)", RFC 6691, July 2012.

[RFC6706] Templin, F., "Asymmetric Extended Route Optimization (AERO)", RFC 6706, August 2012.

[RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field", RFC 6864, February 2013.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, April 2013.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, April 2013.

[RFC6939] Halwasia, G., Bhandari, S., and W. Dec, "Client Link-Layer Address Option in DHCPv6", RFC 6939, May 2013.

[RFC6980] Gont, F., "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery", RFC 6980, August 2013.

[RFC7078] Matsumoto, A., Fujisaki, T., and T. Chown, "Distributing Address Selection Policy Using DHCPv6", RFC 7078, January 2014.

Author's Address

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

Email: fltemplin@acm.org