Network Working Group
Internet-Draft
Obsoletes: rfc6706 (if approved)
Intended status: Standards Track
Expires: October 31, 2014

F. Templin, Ed.
Boeing Research & Technology

April 29, 2014

Transmission of IPv6 Packets over AERO Links
draft-templin-aerolink-16.txt

Abstract

This document specifies the operation of IPv6 over tunnel virtual Non-Broadcast, Multiple Access (NBMA) links using Asymmetric Extended Route Optimization (AERO). Nodes attached to AERO links can exchange packets via trusted intermediate routers on the link that provide forwarding services to reach off-link destinations and/or redirection services to inform the node of an on-link neighbor that is closer to the final destination. Operation of the IPv6 Neighbor Discovery (ND) protocol over AERO links is based on an IPv6 link local address format known as the AERO address.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This document specifies the operation of IPv6 over tunnel virtual Non-Broadcast, Multiple Access (NBMA) links using Asymmetric Extended Route Optimization (AERO). Nodes attached to AERO links can exchange packets via trusted intermediate routers on the link that provide forwarding services to reach off-link destinations and/or redirection services to inform the node of an on-link neighbor that is closer to the final destination.

Nodes on AERO links use an IPv6 link-local address format known as the AERO Address. This address type has properties that statelessly link IPv6 Neighbor Discovery (ND) to IPv6 routing. The AERO link can be used for tunneling to neighboring nodes on either IPv6 or IPv4 networks, i.e., AERO views the IPv6 and IPv4 networks as equivalent links for tunneling. 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 IPv6.

AERO interface
   a node's attachment to an AERO link. The AERO interface Maximum Transmission Unit (MTU) is less than or equal to the AERO link MTU.

AERO address
   an IPv6 link-local address assigned to an AERO interface and constructed as specified in Section 3.6.

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

AERO Client ("client")
   a node that configures either a host interface or a router interface on an AERO link.

AERO Server ("server")
a node that configures a router interface on an AERO link over which it can provide default forwarding and redirection services for other AERO nodes.

AERO Relay ("relay")
a node that relays IPv6 packets between Servers on the same AERO link, and/or that forwards IPv6 packets between the AERO link and the IPv6 Internet. An AERO Relay may or may not also be configured as an AERO 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 AERO nodes tunnel IPv6 packets.

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

underlying address
an IPv6 or IPv4 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 underlying address. Underlying addresses are used as the source and destination addresses of the AERO encapsulation header.

link-layer address
the same as defined for "underlying address" above.

network layer address
an IPv6 address used as the source or destination address of the inner IPv6 packet header.

end user network (EUN)
an IPv6 network attached to a downstream interface of an AERO Client (where the AERO interface is seen as the upstream interface).

Throughout the document, the simple terms "Server" and "Relay" refer to "AERO Server" and "AERO Relay", respectively. Capitalization is

used to distinguish these terms from DHCPv6 server and DHCPv6 relay. This is an important distinction, since an AERO Server may be a DHCPv6 relay, and an AERO Relay may be a DHCPv6 server.

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 IPv6 over Asymmetric Extended Route Optimization (AERO) links:

3.1. AERO Node Types

AERO Relays relay packets between nodes connected to the same AERO link and also forward packets between the AERO link and the native IPv6 network. The relaying process entails re-encapsulation of IPv6 packets that were received from a first AERO node and are to be forwarded without modification to a second AERO node.

AERO Servers configure their AERO interfaces as router interfaces, and provide default routing services to AERO Clients. AERO Servers configure a DHCPv6 relay or server function and facilitate DHCPv6 Prefix Delegation (PD) exchanges. An AERO Server may also act as an AERO Relay.

AERO Clients act as requesting routers to receive IPv6 prefixes through a DHCPv6 PD exchange via an AERO Server over the AERO link. Each AERO Client receives at least a /64 prefix delegation, and may receive even shorter prefixes.

AERO Clients that act as routers configure their AERO interfaces as router interfaces, i.e., even if the AERO Client otherwise displays the outward characteristics of an ordinary host (for example, the Client may internally configure both an AERO interface and (internal virtual) End User Network (EUN) interfaces). AERO Clients that act as routers sub-delegate portions of their received prefix delegations to links on EUNs.

AERO Clients that act as ordinary hosts configure their AERO interfaces as host interfaces and assign one or more IPv6 addresses taken from their received prefix delegations to the AERO interface but DO NOT assign the delegated prefix itself to the AERO interface. Instead, the host assigns the delegated prefix to a "black hole" route so that unused portions of the prefix are nullified.

End system applications on AERO hosts bind directly to the AERO interface, while applications on AERO routers (or IPv6 hosts served by an AERO router) bind to EUN interfaces.

3.2. AERO Interface Characteristics

AERO interfaces use IPv6-in-IPv6 encapsulation [RFC2473] to exchange tunneled packets with AERO neighbors attached to an underlying IPv6 network, and use IPv6-in-IPv4 encapsulation [RFC4213] to exchange tunneled packets with AERO neighbors attached to an underlying IPv4 network. AERO interfaces can also use IPsec encapsulation [RFC4301] (either IPv6-in-IPsec-in-IPv6 or IPv6-in-IPsec-in-IPv4) in environments where strong authentication and confidentiality are required. When NAT traversal and/or filtering middlebox traversal is necessary, a UDP header is further inserted between the outer IP encapsulation header and the inner packet.

Servers assign the link-local address 'fe80::0' to their AERO interface; this provides a handle for Clients to insert into a neighbor cache entry for their current Server. Servers also configure administratively-assigned link-local addresses on their AERO interfaces to support the operation of the inter-Server/Relay routing system (see: [IRON]).

Clients initially assign no addresses on their AERO interface, but use 'fe80::1' as the IPv6 link-local address in the DHCPv6 PD exchanges used to receive an IPv6 prefix and derive an AERO address. After the Client receives a prefix delegation, it assigns the corresponding AERO address to the AERO interface.

AERO interfaces maintain a neighbor cache and use a variation of standard unicast IPv6 ND messaging. AERO interfaces use Neighbor Solicitation (NS), Neighbor Advertisement (NA) and Redirect messages the same as for any IPv6 link. They do not use Router Solicitation (RS) and Router Advertisement (RA) messages for several reasons. First, default router discovery is supported through other means more appropriate for AERO links as described below. Second, discovery of more-specific routes is through the receipt of Redirect messages. Finally, AERO nodes obtain their delegated IPv6 prefixes using DHCPv6 PD; hence, there is no need for RA-based prefix discovery.

AERO Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages do not include Source/Target Link Layer Address Options (S/TLLAO). Instead, AERO nodes determine the link-layer addresses of neighbors by examining the encapsulation IP source address and UDP port number (when UDP encapsulation is used) of any NS/NA messages they receive and ignore any S/TLLAOs included in these messages. This is vital to the operation of AERO links for which neighbors are

separated by Network Address Translators (NATs) - either IPv4 or IPv6.

AERO Redirect messages include a TLLAO the same as for any IPv6 link. The TLLAO includes the link-layer address of the target node, including both the IP address and the UDP source port number used by the target when it sends UDP-encapsulated packets over the AERO interface (the TLLAO instead encodes the value 0 when the target does not use UDP encapsulation). TLLAOs for target nodes that use an IPv6 underlying address include the full 16 bytes of the IPv6 address as shown in Figure 1, while TLLAOs for target nodes that use an IPv4 underlying address include only the 4 bytes of the IPv4 address as shown in Figure 2.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |    Type = 2   |   Length = 3  |     UDP Source Port (or 0)    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                           Reserved                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +--                                                           --+
    |                                                               |
    +--                        IPv6 Address                       --+
    |                                                               |
    +--                                                           --+
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: AERO TLLAO Format for IPv6

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |    Type = 2   |   Length = 1  |     UDP Source Port (or 0)    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          IPv4 Address                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: AERO TLLAO Format for IPv4

Finally, AERO interface NS/NA messages only update existing neighbor cache entires and do not create new neighbor cache entries, whereas Redirect messages both update and create neighbor cache entries. This represents a departure from the normal operation of IPv6 ND over

common link types, but is consistent with the spirit of IPv6 over NBMA links as discussed in [RFC4861]. Note however that this restriction may be relaxed and/or redefined on AERO links that participate in a fully distributed mobility management model coordinated in a manner outside the scope of this document.

3.3. AERO Addresses

An AERO address is an IPv6 link-local address assigned to an AERO interface and with an IPv6 prefix embedded within the interface identifier. The AERO address is formatted as:

```
fe80::[IPv6 prefix]
```

Each AERO Client configures an AERO address based on the delegated prefix it has received from the DHCPv6 server. The address begins with the prefix fe80::/64 and includes in its interface identifier the base /64 prefix taken from the Client's delegated IPv6 prefix. The base prefix is determined by masking the delegated prefix with the prefix length. For example, if an AERO Client has received the prefix delegation:

```
2001:db8:1000:2000::/56
```

it would construct its AERO address as:

```
fe80::2001:db8:1000:2000
```

The AERO address remains stable as the Client moves between topological locations, i.e., even if its underlying address changes.

3.4. AERO Interface Data Origin Authentication

Nodes on AERO interfaces use a simple data origin authentication for encapsulated packets they receive from other nodes. In particular, AERO Clients accept encapsulated packets with a link-layer source address belonging to their current AERO Server. AERO nodes also accept encapsulated packets with a link-layer source address that is correct for the network-layer source address.

The AERO node considers the link-layer source address correct for the network-layer source address if there is an IPv6 forwarding table entry that matches the network-layer source address as well as a neighbor cache entry corresponding to the next hop that includes the link-layer address. An exception is that NS, NA and Redirect messages may include a different link-layer address than the one currently in the neighbor cache, and the new link-layer address updates the neighbor cache entry.

3.5. AERO Interface Conceptual Data Structures and Protocol Constants

Each AERO node maintains a per-AERO interface conceptual neighbor cache that includes an entry for each neighbor it communicates with on the AERO link, the same as for any IPv6 interface (see [RFC4861]). Neighbor cache entries are either static or dynamic. Static neighbor cache entries (including a Client's neighbor cache entry for a Server or a Server's neighbor cache entry for a Client) do not have timeout values and are retained until explicitly deleted. Dynamic neighbor cache entries are created and maintained by the AERO redirection procedures describe in the following sections.

When an AERO node receives a valid Predirect message (See Section 3.11.5) it creates or updates a dynamic neighbor cache entry for the Predirect target L3 and L2 addresses, and also creates an IPv6 forwarding table entry for the Predirected (source) prefix. The node then sets an ACCEPT timer and uses this timer to validate any messages received from the Predirected neighbor.

When an AERO node receives a valid Redirect message (see Section 3.11.7) it creates or updates a dynamic neighbor cache entry for the Redirect target L3 and L2 addresses, and also creates an IPv6 forwarding table entry for the Redirected (destination) prefix. The node then sets a FORWARD timer and uses this timer to determine whether packets can be sent directly to the Redirected neighbor. The node also maintains a constant value MAX_RETRY to limit the number of keepalives sent when a neighbor has gone unreachable.

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 neighbor discovery [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 the ACCEPT_TIME timer decrements below FORWARD_TIME.

It is RECOMMENDED that MAX_RETRY be set to 3 the same as described for IPv6 neighbor discovery 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. ACCEPT_TIME SHOULD further be set to a value that is sufficiently longer than FORWARD_TIME to allow the AERO redirection procedure to converge.

3.6. AERO Interface MTU Considerations

The AERO link Maximum Transmission Unit (MTU) is 64KB minus the encapsulation overhead for IPv4 [RFC0791] and 4GB minus the encapsulation overhead for IPv6 [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 such as IPsec 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 1500 bytes and the underlying interface MTU minus the encapsulation overhead. 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 outer IP fragmentation to fragment the packet into two pieces (where the first fragment contains 1024 bytes of the fragmented inner packet) then admits the fragments into the tunnel. If the outer protocol is IPv4, the node admits the packet 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

ICMPv6 Neighbor Solicitation (NS) message with trailing padding octets added to a length of 1500 bytes but does not include the length of the padding in the IPv6 Payload Length field. The node then encapsulates the NS in the outer encapsulation headers (while including the length of the padding in the outer length fields), sets DF to 1 (for IPv4) and sends the padded NS message to the neighbor. If the neighbor returns an NA message, the node may then send whole packets within this size range and (for IPv4) relax the rate limiting requirement.

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 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.

3.7. AERO Interface Encapsulation, Re-encapsulation and Decapsulation

AERO interfaces encapsulate IPv6 packets according to whether they are entering the AERO interface for the first time or if they are being forwarded out the same AERO interface that they arrived on. This latter form of encapsulation is known as "re-encapsulation".

AERO interfaces encapsulate packets per the specifications in , [RFC2473], [RFC4213] except that the interface copies the "TTL/Hop Limit", "Type of Service/Traffic Class" and "Congestion Experienced" values in the inner network layer header into the corresponding fields in the outer IP 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 outer IP header into the corresponding fields in the new outer IP header (i.e., the values are transferred between outer headers and *not* copied from the inner network layer header).

When UDP encapsulation is used, the AERO interface inserts a UDP header between the inner packet and outer IP header. If the outer header is IPv6 and is followed by an IPv6 Fragment header, the AERO interface inserts the UDP header between the outer IPv6 header and IPv6 Fragment 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 inner 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-registerd port number for AERO). 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 outer IP protocol number to the appropriate value for the first protocol layer within the encapsulation (e.g., IPv6, IPv6 Fragment Header, UDP, etc.). When IPv6 is used as the outer IP protocol, the ITE then sets the flow label value in the outer IPv6 header the same as described in [RFC6438]. When IPv4 is used as the outer IP protocol, the AERO interface sets the DF bit as discussed in Section 3.2.

AERO interfaces decapsulate packets destined either to the node itself or to a destination reached via an interface other than the receiving AERO interface per the specifications in , [RFC2473], [RFC4213]. When the encapsulated packet includes a UDP header, the AERO interface examines the first octet of data following the UDP header to determine the inner header type. If the most significant four bits of the first octet encode the value '0110', the inner header is an IPv6 header. Otherwise, the interface considers the first octet as an IP protocol type that encodes the value '44' for IPv6 fragment header, the value '50' for Encapsulating Security Payload, the value '51' for Authentication Header etc. (If the first octet encodes the value '0', the interface instead discards the packet, since this is the value reserved for experimentation under , [RFC6706]). During the decapsulation, the AERO interface records the UDP source port in the neighbor cache entry for this neighbor then discards the UDP header.

3.8. AERO Reference Operational Scenario

Figure 3 depicts the AERO reference operational scenario. The figure shows an AERO Server('A'), two AERO Clients ('B', 'D') and three ordinary IPv6 hosts ('C', 'E', 'F'):

```
                    .-(::::::::)
                 .-(::: IPv6 :::)-.   +-------------+
                (:::: Internet ::::)--|    Host F   |
                 `-(::::::::::::)-'   +-------------+
                    `-(::::::)-'       2001:db8:3::1
                         |
                  +--------------+
                  | AERO Server A|
                  | (C->B; E->D) |
                  +--------------+
                      fe80::0
                       L2(A)
                         |
       X-----+-----------+-----------+--------X
             |       AERO Link       |
            L2(B)                  L2(D)
     fe80::2001:db8:0:0      fe80::2001:db8:1:0         .-.
     +--------------+         +--------------+       ,-(  _)-.
     | AERO Client B|         | AERO Client D|    .-(_ IPv6  )-.
     | (default->A) |         | (default->A) |--(__    EUN      )
     +--------------+         +--------------+     `-(______)-'
     2001:DB8:0::/48           2001:DB8:1::/48           |
             |                                     2001:db8:1::1
            .-.                                   +-------------+
         ,-(  _)-.      2001:db8:0::1             |    Host E   |
      .-(_ IPv6  )-.   +-------------+            +-------------+
    (__    EUN      )--|    Host C   |
       `-(______)-'    +-------------+
```

Figure 3: AERO Reference Operational Scenario

In Figure 3, AERO Server ('A') connects to the AERO link and connects to the IPv6 Internet, either directly or via an AERO Relay (not shown). Server ('A') assigns the address fe80::0 to its AERO interface with link-layer address L2(A). Server ('A') next arranges to add L2(A) to a published list of valid Servers for the AERO link.

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

AERO Client ('D') registers the IPv6 prefix 2001:db8:1::/48 in a DHCPv6 PD exchange via AERO Server ('A') then assigns the address

fe80::2001:db8:1:0 to its AERO interface with link-layer address L2(D). Client ('D') configures a default route via the AERO interface with next-hop address fe80::0 and link-layer address L2(A), then sub-delegates the prefix 2001:db8:1::/48 to its attached EUNs. IPv6 host ('E') connects to the EUN, and configures the address 2001:db8:1::1.

Finally, IPv6 host ('F') connects to an IPv6 network outside of the AERO link domain. Host ('F') configures its IPv6 interface in a manner specific to its attached IPv6 link, and assigns the address 2001:db8:3::1 to its IPv6 link interface.

3.9. AERO Router Discovery and Prefix Delegation

3.9.1. AERO Client Behavior

AERO Clients observe the IPv6 node requirements defined in [RFC6434]. AERO Clients first discover the link-layer address of an AERO Server 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 "domainname" is the DNS domain appropriate for the Client's attached underlying network. The Client then creates a static neighbor cache entry with fe80::0 as the network-layer address and the discovered address as the link-layer address. The Client further creates a static default IPv6 forwarding table entry with fe80::0 as the next hop address.

Next, the Client acts as a requesting router to request an IPv6 prefix through DHCPv6 PD [RFC3633] via the AERO Server using fe80::1 as the IPv6 source address and fe80::0 as the IPv6 destination address. The Client further includes a DHCPv6 Unique Identifier (DUID) based on a Universally Unique Identifier (UUID) (also known as DUID-UUID) as described in [RFC6355].

After the Client receives its prefix delegation, it assigns the link-local AERO address taken from the prefix to the AERO interface (see: Section 3.3) and sub-delegates the prefix to nodes and links within its attached EUNs (the AERO link-local address thereafter remains stable as the Client moves). The Client further renews its prefix delegation via standard DHCPv6 procedures by sending DHCPv6 Renew messages with fe80::1 as the IPv6 source address, fe80::0 as the IPv6 destination address and the same DUID-UUID value as the DUID.

The Client sends periodic NS messages to the Server to obtain new NAs in order to refresh any network state. The Client can also forward IPv6 packets destined to networks beyond its local EUNs via the Server as an IPv6 default router. The Server may in turn return a

Redirect message informing the Client of a neighbor on the AERO link that is topologically closer to the final destination as specified in Section 3.11.

3.9.2. AERO Server Behavior

AERO Servers observe the IPv6 router requirements defined in [RFC6434] and further configure a DHCPv6 relay or server function on their AERO links. When the AERO Server relays a Client's DHCPv6 PD messages to the DHCPv6 server, it wraps each message in a "Relay-forward" message per [RFC3315] and includes a DHCPv6 "Interface-Id" option that encodes a value that identifies the AERO link to the DHCPv6 server. The AERO Server then encodes the Client's link-layer source address and (when UDP encapsulation is used) UDP source port number in the link-address and peer-address fields of the "Relay-forward" message. (Note that the link-address and peer-address fields of the "Relay-forward" message are available for coding since the message includes an "Interface-Id" option (thereby obviating the need for a link-address) and the source address of the DHCPv6 message will always be fe80::1 (thereby obviating the need for a peer-address). Most importantly, encoding the link-layer information in this manner allows the AERO Server to remain stateless while the DHCPv6 server is authoritative for the delegation of IPv6 prefixes to authorized AERO Clients.)

When the DHCPv6 server issues the IPv6 prefix delegation in a "Relay-reply" message via the AERO Server (acting as a DHCPv6 relay), the AERO Server creates a static neighbor cache entry for the Client's AERO address (see: Section 3.3) with the coded link-layer address and UDP port number as the link-layer address for the neighbor cache entry. The AERO Server also configures a static IPv6 forwarding table entry that lists the Client's AERO address as the next hop toward the delegated IPv6 prefix .The AERO Server finally injects the Client's prefix as an IPv6 route into the inter-Server/Relay routing system (see: [IRON]) then relays the DHCPv6 message to the Client while using fe80::0 as the IPv6 source address, fe80::1 as the IPv6 destination address, and the Client's cached link-layer address as the destination link-layer address.

Servers respond to NS messages from Clients on their AERO interfaces by returning an NA message. When the Server receives an NS message, it updates the neighbor cache entry using the network layer source address as the neighbor's network layer address and using the link-layer source address of the NS message as the neighbor's link-layer address.

When the Server forwards a packet via the same AERO interface on which it arrived, it initiates an AERO route optimization procedure

as specified in Section 3.11.

3.10. AERO Neighbor Solicitation and Advertisement

Each AERO node uses its delegated prefix to create an AERO address (see: Section 3.3). It can then send unicast NS messages to elicit NA messages from other AERO nodes the same as described for neighbor unreachability detection in[RFC4861] except that the messages do not include S/TLLAOs.

When an AERO node sends an NS/NA message, it MUST use its AERO address as the IPv6 source address and the AERO address of the neighbor as the IPv6 destination address. The AERO node also includes the AERO address of the neighbor as the NS/NA message Target address with the exception of "terminating NS" messages as described in Section 3.13.

When an AERO node receives an NS/NA message, it accepts the message if it has a neighbor cache entry for the neighbor; otherwise, it ignores the message.

3.11. AERO Redirection

Section 3.8 describes the AERO reference operational scenario. We now discuss the operation and protocol details of AERO Redirection with respect to this reference scenario.

3.11.1. Classical Redirection Approaches

With reference to Figure 3, when the IPv6 source host ('C') sends a packet to an IPv6 destination host ('E'), the packet is first forwarded via the EUN to AERO Client ('B'). Client ('B') then forwards the packet over its AERO interface to AERO Server ('A'), which then re-encapsulates and forwards the packet to AERO Client ('D'), where the packet is finally forwarded to the IPv6 destination host ('E'). When Server ('A') re-encapsulates and forwards the packet back out on its advertising AERO interface, it must arrange to redirect Client ('B') toward Client ('D') as a better next-hop node on the AERO link that is closer to the final destination. However, this redirection process applied to AERO interfaces must be more carefully orchestrated than on ordinary links since the parties may be separated by potentially many underlying network routing hops.

Consider a first alternative in which Server ('A') informs Client ('B') only and does not inform Client ('D') (i.e., "classical redirection"). In that case, Client ('D') has no way of knowing that Client ('B') is authorized to forward packets from the claimed source address, and it may simply elect to drop the packets. Also, Client

('B') has no way of knowing whether Client ('D') is performing some form of source address filtering that would reject packets arriving from a node other than a trusted default router, nor whether Client ('D') is even reachable via a direct path that does not involve Server ('A').

Consider a second alternative in which Server ('A') informs both Client ('B') and Client ('D') separately, via independent redirection control messages (i.e., "augmented redirection"). In that case, if Client ('B') receives the redirection control message but Client ('D') does not, subsequent packets sent by Client ('B') could be dropped due to filtering since Client ('D') would not have a route to verify the claimed source address. Also, if Client ('D') receives the redirection control message but Client ('B') does not, subsequent packets sent in the reverse direction by Client ('D') would be lost.

Since both of these alternatives have shortcomings, a new redirection technique (i.e., "AERO redirection") is needed.

3.11.2. AERO Redirection Concept of Operations

Again, with reference to Figure 3, when source host ('C') sends a packet to destination host ('E'), the packet is first forwarded over the source host's attached EUN to Client ('B'), which then forwards the packet via its AERO interface to Server ('A').

Server ('A') then re-encapsulates and forwards the packet out the same AERO interface toward Client ('D') and also sends an AERO "Predirect" message forward to Client ('D') as specified in Section 3.11.4. The Predirect message includes Client ('B')'s network- and link-layer addresses as well as information that Client ('D') can use to determine the IPv6 prefix used by Client ('B') . After Client ('D') receives the Predirect message, it process the message and returns an AERO Redirect message destined for Client ('B') via Server ('A') as specified in Section 3.11.5. During the process, Client ('D') also creates or updates a dynamic neighbor cache entry for Client ('B'), and creates an IPv6 forwarding table entry for Client ('B')'s IPv6 prefix.

When Server ('A') receives the Redirect message, it re-encapsulates the message and forwards it on to Client ('B') as specified in Section 3.11.6. The message includes Client ('D')'s network- and link-layer addresses as well as information that Client ('B') can use to determine the IPv6 prefix used by Client ('D'). After Client ('B') receives the Redirect message, it processes the message as specified in Section 3.11.7. During the process, Client ('B') also creates or updates a dynamic neighbor cache entry for Client ('D'), and creates an IPv6 forwarding table entry for Client ('D')'s IPv6

prefix.

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

3.11.3. AERO Redirection 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 (valid values for the Prefix Length field are 0 through 64). The Redirect/Predirect messages are formatted as shown in Figure 4:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Type (=137)  |  Code (=0/1)  |          Checksum             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                  Reserved                     | Prefix Length |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                                                               +
    |                                                               |
    +                       Target Address                          +
    |                                                               |
    +                                                               +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                                                               +
    |                                                               |
    +                     Destination Address                       +
    |                                                               |
    +                                                               +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Options ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 4: AERO Redirect/Predirect Message Format

3.11.4. Sending Predirects

When an AERO Server forwards a packet out the same AERO interface that it arrived on, the Server sends a Predirect message forward toward the AERO Client nearest the destination instead of sending a

Redirect message back to AERO Client nearest the source.

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

- o the link-layer source address is set to 'L2(A)' (i.e., the underlying address of Server ('A')).
- o the link-layer destination address is set to 'L2(D)' (i.e., the underlying address of Client ('D')).
- o the network-layer source address is set to fe80::0 (i.e., the link-local address of Server ('A')).
- o the network-layer destination address is set to fe80::2001:db8:1:0 (i.e., the AERO address of Client ('D')).
- o the Type is set to 137.
- o the Code is set to 1 to indicate "Predirect".
- o the Prefix Length is set to the length of the prefix to be applied to Target address.
- o the Target Address is set to fe80::2001:db8:0::0 (i.e., the AERO address of Client ('B')).
- o the Destination Address is set to the IPv6 source address of the packet that triggered the Predirection event.
- o the message includes a TLLAO set to 'L2(B)' (i.e., the underlying address of Client ('B')).
- o 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.

Server ('A') then sends the message forward to Client ('D').

3.11.5. Processing Predirects and Sending Redirects

When Client ('D') receives a Predirect message, it accepts the message only if it has a link-layer source address of the Server, i.e. 'L2(A)'. Client ('D') further accepts the message only if it

is willing to serve as a redirection target. Next, Client ('D') validates the message according to the ICMPv6 Redirect message validation rules in Section 8.1 of [RFC4861].

In the reference operational scenario, when the Client ('D') 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 ('B') and stores the link-layer address found in the TLLAO as the link-layer address of Client ('B'). Client ('D') then applies the Prefix Length to the Interface Identifier portion of the Target Address and records the resulting IPv6 prefix in its IPv6 forwarding table.

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

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

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

o the network-layer source address is set to 'L3(D)' (i.e., the AERO address of Client ('D')).

o the network-layer destination address is set to 'L3(B)' (i.e., the AERO address of Client ('B')).

o the Type is set to 137.

o the Code is set to 0 to indicate "Redirect".

o the Prefix Length is set to the length of the prefix to be applied to the Target and Destination address.

o the Target Address is set to fe80::2001:db8:1::1 (i.e., the AERO address of Client ('D')).

o the Destination Address is set to the IPv6 destination address of the packet that triggered the Redirection event.

o the message includes a TLLAO set to 'L2(D)' (i.e., the underlying address of Client ('D')).

o 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 ('D') prepares the Redirect message, it sends the message to Server ('A').

3.11.6. Re-encapsulating and Relaying Redirects

When Server ('A') receives a Redirect message, it accepts the message only if it has a neighbor cache entry that associates the message's link-layer source address with the network-layer source address. Next, Server ('A') validates the message according to the ICMPv6 Redirect message validation rules in Section 8.1 of [RFC4861]. Following validation, Server ('A') re-encapsulates the Redirect then relays the re-encapsulated Redirect on to Client ('B') as follows.

In the reference operational scenario, Server ('A') receives the Redirect message from Client ('D') and prepares to re-encapsulate and forward the message to Client ('B'). Server ('A') first verifies that Client ('D') is authorized to use the Prefix Length in the Redirect message when applied to the AERO address in the network-layer source of the Redirect message, and discards the message if verification fails. Otherwise, Server ('A') re-encapsulates the message by changing the link-layer source address of the message to 'L2(A)', changing the network-layer source address of the message to fe80::0, and changing the link-layer destination address to 'L2(B)' . Server ('A') finally relays the re-encapsulated message to the ingress node ('B') without decrementing the network-layer IPv6 header Hop Limit field.

While not shown in Figure 3, AERO Relays relay Redirect and Predirect messages in exactly this same fashion described above. See Figure 5 in Appendix A for an extension of the reference operational scenario that includes Relays.

3.11.7. Processing Redirects

When Client ('B') receives the Redirect message, it accepts the message only if it has a link-layer source address of the Server, i.e. 'L2(A)'. Next, Client ('B') validates the message according to the ICMPv6 Redirect message validation rules in Section 8.1 of [RFC4861]. Following validation, Client ('B') then processes the message as follows.

In the reference operational scenario, when Client ('B') 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 ('D') and stores the link-layer address found in the TLLAO as the link-layer address of Client ('D'). Client ('B') then applies the Prefix Length to the Interface Identifier portion of the Target Address and records the resulting

IPv6 prefix in its IPv6 forwarding table.

Now, Client ('B') has an IPv6 forwarding table entry for Client('D')'s prefix, and Client ('D') has an IPv6 forwarding table entry for Client ('B')'s prefix. Thereafter, the clients may exchange ordinary network-layer data packets directly without forwarding through Server ('A').

3.12. Neighbor Reachability Maintenance

When a source Client is redirected to a target Client it MUST test the direct path to the target by sending an initial NS message to elicit a solicited NA response. While testing the path, the source Client SHOULD continue sending packets via the Server until target Client reachability has been confirmed. The source Client MUST thereafter continue to test the direct path to the target Client (see Section 7.3 of [RFC4861]) in order to keep dynamic neighbor cache entries alive. In particular, the source Client sends NS messages to the target Client subject to rate limiting in order to receive solicited NA messages. If at any time the direct path appears to be failing, the source Client can resume sending packets via the Server which may or may not result in a new redirection event.

When a target Client receives an NS message from a source Client, it resets the ACCEPT_TIME timer if a neighbor cache entry exists; otherwise, it discards the NS message.

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

When both the FORWARD_TIME and ACCEPT_TIME timers on a dynamic neighbor cache entry expire, the Client deletes both the neighbor cache entry and the corresponding IPv6 forwarding table entry.

If the source Client is unable to elicit an NA response from the target Client after MAX_RETRY attempts, it SHOULD consider the direct path unusable for forwarding purposes. Otherwise, the source Client may continue to send packets directly to the target Client 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.

3.13. Mobility and Link-Layer Address Change Considerations

When a Client needs to change its link-layer address (e.g., due to a mobility event, due to a change in underlying network interface, etc.), it sends an immediate NS message forward to any of its

correspondents (including the Server and any other Clients) which then discover the new link-layer address.

If two Clients change their link-layer addresses simultaneously, the NS/NA messages may be lost. In that case, the Clients SHOULD delete their respective dynamic neighbor cache and IPv6 forwarding table entries, and allow packets to again flow through their Server(s) which MAY result in a new AERO redirection exchange.

When a Client needs to change to a new AERO Server, it issues a new DHCPv6 Request message via the new AERO Server as the DHCPv6 relay. The new AERO Server then relays the message to the DHCPv6 server and processes the resulting exchange the same as described in Section 3.9.2. After the Client receives the resulting DHCPv6 Reply message, it changes the link-layer address of the neighbor cache entry for fe80::0 to the address of the new AERO Server.

After conducting the above exchange via the new AERO Server, the Client then sends a "terminating NS" message to the old AERO Server. The terminating NS message is prepared exactly the same as for an ordinary NS message, except that the Code field contains the value '1'. When the old Server receives the terminating NS message, it withdraws the IPv6 route from the routing system and deletes the neighbor cache entry and IPv6 forwarding table entry for the Client. The old Server then returns an NA message which the Client can use to verify that the termination signal has been processed. (Note that the old Server can impose a small delay before deleting the neighbor cache and IPv6 forwarding table entries so that any packets already in the system can still be delivered to the Client.)

An alternative to sending a "terminating NS" message would be for the Client to somehow perform a DHCPv6 exchange with the DHCPv6 relay function on the old AERO Server but without involving the DHCPv6 server. This would be insecure because the Client only has a DHCPv6 security association with the DHCPv6 server and not the DHCPv6 relay. Conversely, the AERO Client and Server already require an authentic exchange of IPv6 Neighbor Discovery messages.

3.14. Underlying 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 source Client has no means for reaching the target directly (since they connect to underlying networks of different IP protocol versions) and so must ignore any Redirects and continue to send packets via the Server.

3.15. 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 underlying IP address of the AERO 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 inner packet is "M", then the IPv6 multicast destination address of the encapsulating header is also "M".)

3.16. Operation on Server-less AERO Links

In some AERO link scenarios, there may be no Server on the link and/or no need for Clients to use a Server as an intermediary trust anchor. In that case, Clients can establish dynamic neighbor cache entries and IPv6 forwarding table entries by performing direct Client-to-Client Predirect/Redirect 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 prefix.

When there is no Server on the link, Clients must arrange to receive prefix delegations and publish the delegations via a secure prefix discovery service through some means outside the scope of this document.

3.17. Other Considerations

IPv6 hosts serviced by an AERO Client can reach IPv4-only services via a NAT64 gateway [RFC6146] within the IPv6 network.

AERO nodes can use the Default Address Selection Policy with DHCPv6 option [RFC7078] the same as on any IPv6 link.

All other (non-multicast) functions that operate over ordinary IPv6 links operate in the same fashion over AERO links.

4. Implementation Status

An early implementation is available at: http://linkupnetworks.com/aero/aerov2-0.1.tgz.

5. IANA Considerations

This document uses the UDP Service Port Number 8060 reserved by IANA for AERO in [RFC6706]. Therefore, there are no new IANA actions required for this document.

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 is dependent on a trust basis between AERO Clients and Servers, where the Clients only engage in the AERO mechanism when it is facilitated by a trust anchor.

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., WiFi networks) and links that provide physical security (e.g., enterprise network LANs) provide a first line of defense that is often sufficient. In other instances, securing mechanisms such as Secure Neighbor Discovery (SeND) [RFC3971] or IPsec [RFC4301] may be necessary.

AERO Clients MUST ensure that their connectivity is not used by unauthorized nodes to gain access to a protected network. (This concern is no different than for ordinary hosts that receive an IP address delegation but then "share" the address with unauthorized nodes via an IPv6/IPv6 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].

7. Acknowledgements

Discussions both on the v6ops list 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, Brian Haberman, Joel Halpern, Sascha Hlusiak, Lee Howard and Joe Touch. 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, Balaguruna Chidambaram, Jeff

Holland, Cam Brodie, Yueli Yang, Wen Fang, Ed King, Mike Slane, Kent Shuey, Gen MacLean, 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.

[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.

[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.

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

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

8.2. Informative References

[IRON] Templin, F., "The Internet Routing Overlay Network (IRON)", Work in Progress, June 2012.

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

[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.

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

[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.

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

[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.

[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.

[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.

Appendix A. AERO Server and Relay Interworking

Figure 3 depicts a reference AERO operational scenario with a single Server on the AERO link. In order to support scaling to larger numbers of nodes, the AERO link can deploy multiple Servers and Relays, e.g., as shown in Figure 5.

```
                             .-(::::::::)
                          .-(::: IPv6 :::)-.
                         (:: Internetwork ::)
                          `-(::::::::::::)-'
                             `-(::::::)-'
                                  |
       +--------------+    +------+-------+    +--------------+
       |AERO Server C |    | AERO Relay D |    |AERO Server E |
       | (default->D) |    | (A->C; G->E) |    | (default->D) |
       |    (A->B)    |    +-------+------+    |    (G->F)    |
       +-------+------+            |           +------+-------+
               |                   |                  |
       X---+---+-------------------+------------------+---+---X
           |                  AERO Link                   |
     +-----+--------+                            +--------+-----+
     |AERO Client B |                            |AERO Client F |
     | (default->C) |                            | (default->E) |
     +--------------+                            +--------------+
           .-.                                         .-.
        ,-(  _)-.                                   ,-(  _)-.
     .-(_ IPv6  )-.                              .-(_ IPv6  )-.
    (__    EUN      )                           (__    EUN      )
       `-(______)-'                                `-(______)-'
            |                                           |
        +--------+                                  +--------+
        | Host A |                                  | Host G |
        +--------+                                  +--------+
```

Figure 5: AERO Server/Relay Interworking

In this example, AERO Client ('B') associates with AERO Server ('C'), while AERO Client ('F') associates with AERO Server ('E'). Furthermore, AERO Servers ('C') and ('E') do not associate with each other directly, but rather have an association with AERO Relay ('D') (i.e., a router that has full topology information concerning its associated Servers and their Clients). Relay ('D') connects to the AERO link, and also connects to the native IPv6 Internetwork.

When host ('A') sends a packet toward destination host ('G'), IPv6 forwarding directs the packet through the EUN to Client ('B'), which forwards the packet to Server ('C') in absence of more-specific forwarding information. Server ('C') forwards the packet, and it also generates an AERO Predirect message that is then forwarded through Relay ('D') to Server ('E'). When Server ('E') receives the message, it forwards the message to Client ('F').

After processing the AERO Predirect message, Client ('F') sends an AERO Redirect message to Server ('E'). Server ('E'), in turn,

forwards the message through Relay ('D') to Server ('C'). When Server ('C') receives the message, it forwards the message to Client ('B') informing it that host 'G's EUN can be reached via Client ('F'), thus completing the AERO redirection.

The network layer routing information shared between Servers and Relays must be carefully coordinated in a manner outside the scope of this document. In particular, Relays require full topology information, while individual Servers only require partial topology information (i.e., they only need to know the EUN prefixes associated with their current set of Clients). See [IRON] for an architectural discussion of routing coordination between Relays and Servers.

Author's Address

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

Email: fltemplin@acm.org