Network Working Group   F. Templin, Ed.
Internet-Draft   Boeing Research & Technology
Obsoletes: rfc5320, rfc5558, rfc5720,   August 5, 2015
           rfc6179, rfc6706 (if
           approved)
Intended status: Standards Track
Expires: February 6, 2016

                 Asymmetric Extended Route Optimization (AERO)
                         draft-templin-aerolink-61.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, provisioning and mobility
   are supported by the Dynamic Host Configuration Protocol for IPv6
   (DHCPv6), and route optimization is naturally supported through
   dynamic neighbor cache updates. Although DHCPv6 and IPv6 ND
   messaging are used in the control plane, both IPv4 and IPv6 are
   supported in the data plane. AERO is a widely-applicable tunneling
   solution using standard control messaging exchanges as described in
   this document.

Copyright Notice

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, provisioning and mobility are supported by the Dynamic Host
Configuration Protocol for IPv6 (DHCPv6) [RFC3315], and route
optimization is naturally supported through dynamic neighbor cache
updates. Although DHCPv6 and IPv6 ND messaging are used in the
control plane, both IPv4 and IPv6 can be used in the data plane.
AERO is a widely-applicable tunneling solution using standard control
messaging exchanges as described in this document. 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. Nodes typically have a
   single AERO interface; support for multiple AERO interfaces is
   also possible but out of scope for this document.

AERO address
   an IPv6 link-local address constructed as specified in Section 3.3
   and assigned to a Client's AERO interface.

AERO node
   a node that is connected to an AERO link and that participates in
   IPv6 ND and DHCPv6 messaging over the link.

AERO Client ("Client")
   a node that issues DHCPv6 messages using the special IPv6 link-
   local address 'fe80::ffff:ffff:ffff:ffff' to receive IP Prefix
   Delegations (PD) from one or more AERO Servers. Following PD, the
   Client assigns an AERO address to the AERO interface which it uses
   in IPv6 ND messaging to coordinate with other AERO nodes.

AERO Server ("Server")
   a node that configures an AERO interface to provide default
   forwarding and DHCPv6 services for AERO Clients. The Server
   assigns an administratively provisioned IPv6 link-local unicast

      address to support the operation of DHCPv6 and the IPv6 ND
      protocol. An AERO Server can also act as an AERO Relay.

   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
      assigns an administratively provisioned IPv6 link-local unicast
      address to the AERO interface the same as for a Server. An AERO
      Relay can also act as an AERO Server.

   AERO Forwarding Agent ("Forwarding Agent")
      a node that performs data plane forwarding services as a companion
      to 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 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 [RFC3315].

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]. Lower case
uses of these words are not to be interpreted as carrying RFC2119
significance.

3. Asymmetric Extended Route Optimization (AERO)

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

3.1. AERO Link Reference Model

```
                           .-(::::::::)
                        .-(::::: IP ::::)-.
                       (:: Internetwork ::)
                        `-(:::::::::::::)-'
                          `-(:::::::)-'
                               |
      +--------------+  +--------+-------+   +--------------+
      |AERO Server S1|  | AERO Relay R1  |   |AERO Server S2|
      |  Nbr: C1; R1 |  |   Nbr: S1; S2  |   |  Nbr: C2; R1 |
      |  default->R1 |  |(H1->S1; H2->S2)|   |  default->R1 |
      |    H1->C1    |  +--------+-------+   |    H2->C2    |
      +-------+------+           |           +------+-------+
              |                  |                  |
      X---+---+------------------+------------------+---+---X
          |           AERO Link                     |
    +-----+--------+                          +--------+-----+
    |AERO Client C1|                          |AERO Client C2|
    |    Nbr: S1   |                          |   Nbr: S2    |
    | default->S1  |                          | default->S2  |
    +--------------+                          +--------------+
          .-.                                       .-.
        ,-(  _)-.                                 ,-(  _)-.
      .-(_   IP  )-.                             .-(_   IP  )-.
     (__    EUN     )                           (__    EUN     )
       `-(_____)-'                               `-(_____)-'
            |                                          |
       +--------+                                 +--------+
       | Host H1|                                 | Host H2|
       +--------+                                 +--------+
```
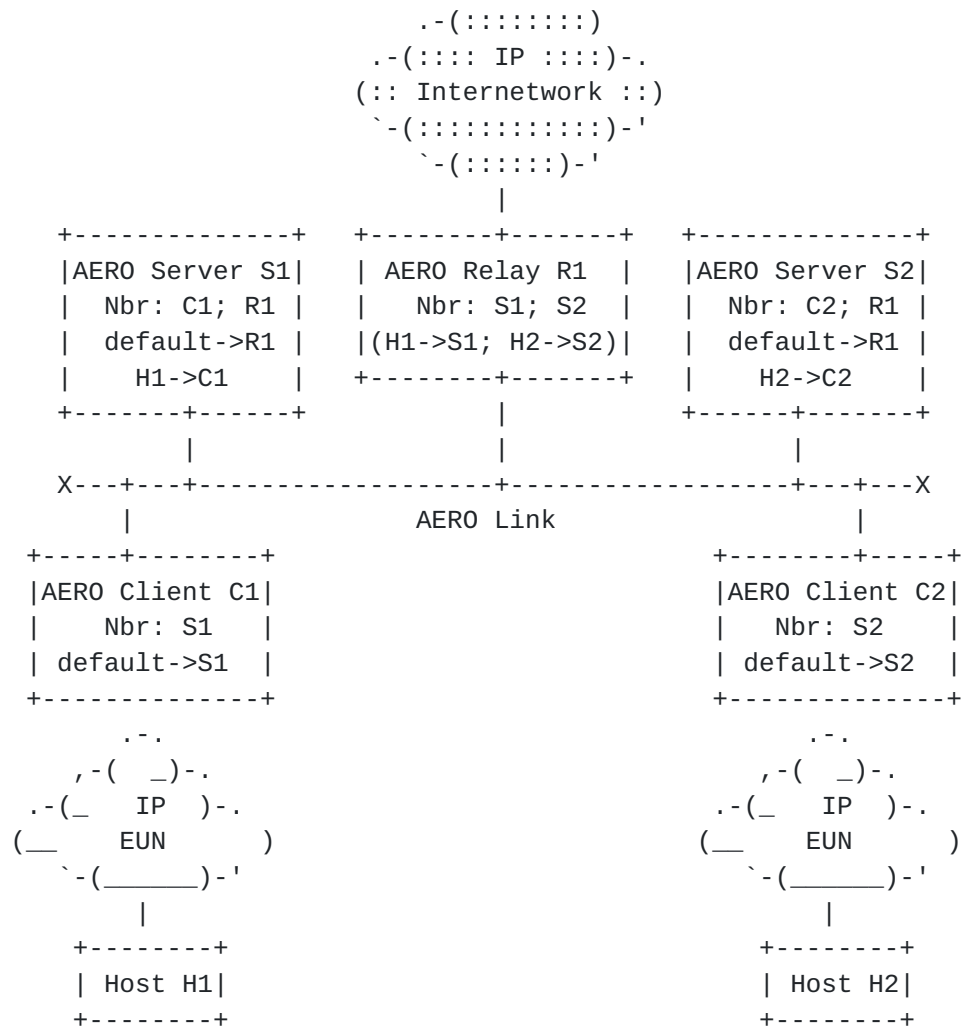
                   Figure 1: AERO Link Reference Model

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

   o Relay R1 acts as a default router for its associated Servers S1
      and S2, and connects the AERO link to the rest of the IP
      Internetwork

   o Servers S1 and S2 associate with Relay R1 and also act as default
      routers for their associated Clients C1 and C2.

   o Clients C1 and C2 associate with Servers S1 and S2, respectively
      and also act as default routers for their associated EUNs

   o Hosts H1 and H2 attach to the EUNs served by Clients C1 and C2,
      respectively

Each node maintains a neighbor cache and IP forwarding table. (For example, AERO Relay R1 in the diagram has neighbor cache entries for Servers S1 and S2 and IP forwarding table entries for the ACPs delegated to Clients C1 and C2.) In common operational practice, there may be many additional Relays, Servers and Clients. (Although not shown in the figure, AERO Forwarding Agents may also be provided for data plane forwarding offload services.)

3.2. AERO Link 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 IP Internetwork. Relays present the AERO link to the native Internetwork as a set of one or more AERO Service Prefixes (ASPs) and serve as a gateway between the AERO link and the Internetwork. AERO Relays maintain an AERO interface neighbor cache entry for each AERO Server, and maintain an IP forwarding table entry for each AERO Client Prefix (ACP). AERO Relays can also be configured to act as AERO Servers.

AERO Servers provide default forwarding services to AERO Clients. Each Server also peers with each Relay in a dynamic routing protocol instance to advertise its list of associated ACPs. Servers configure a DHCPv6 server function to facilitate Prefix Delegation (PD) exchanges with Clients. Each delegated prefix becomes an ACP taken from an ASP. Servers forward packets between AERO interface neighbors only, i.e., and not between the AERO link and the native IP Internetwork. 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 Servers can also be configured to act as AERO Relays.

AERO Clients act as requesting routers to receive ACPs through DHCPv6 PD exchanges with AERO Servers over the AERO link and sub-delegate portions of their ACPs to EUN interfaces. (Each Client MAY associate with a single Server or with multiple Servers, e.g., for fault tolerance, load balancing, etc.) Each IPv6 Client receives at least a /64 IPv6 ACP, and may receive even shorter prefixes. Similarly, each IPv4 Client receives at least a /32 IPv4 ACP (i.e., a singleton IPv4 address), and may receive even shorter prefixes. 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 typically configure a TUN/TAP interface [TUNTAP] as a point-to-point linkage between the IP layer and the AERO interface. The IP layer therefore sees only the TUN/TAP interface, while the

AERO interface provides an intermediate conduit between the TUN/TAP
interface and the underlying interfaces. AERO Clients that act as
hosts assign one or more IP addresses from their ACPs to the TUN/TAP
interface, i.e., and not to the AERO interface.

AERO Forwarding Agents provide data plane forwarding services as
companions to AERO Servers. Note that while Servers are required to
perform both control and data plane operations on their own behalf,
they may optionally enlist the services of special-purpose Forwarding
Agents to offload data plane traffic.

3.3. AERO Addresses

An AERO address is an IPv6 link-local address with an embedded ACP
and assigned 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 advanced
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 encapsulation (see Section 3.10) to exchange
packets with neighbors attached to the AERO link. AERO interfaces
maintain a neighbor cache, and AERO Clients and Servers use 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 3.17). AERO links further use link-local-only
addressing; hence, AERO nodes ignore any Prefix Information Options
(PIOs) they may receive in RA messages over an AERO interface.

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

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    Type = 2   |  Length = 3   |             Reserved          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    Link ID    |    NDSCPs     |  DSCP #1  |Prf|  DSCP #2  |Prf|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  DSCP #3  |Prf|  DSCP #4  |Prf| ....
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       UDP Port Number      |                                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                                 +
   |                                                              |
   +                                                              +
   |                         IP Address                           |
   +                                                              +
   |                                                              |
   +                            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
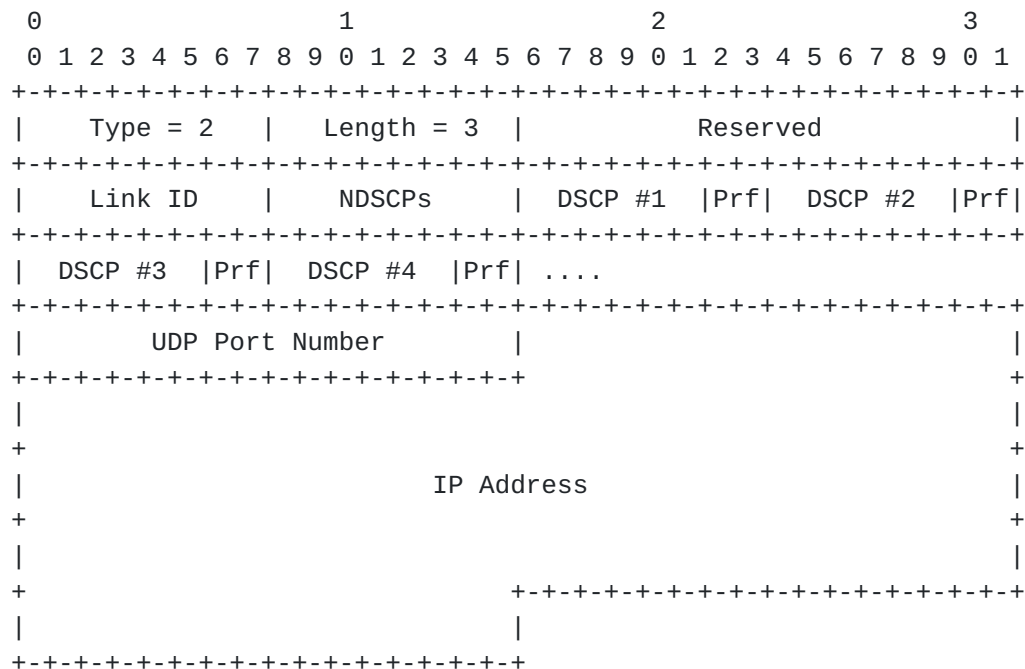
Figure 2: AERO Source/Target Link-Layer Address Option (S/TLLAO)
                            Format

In this format, Link ID is an integer value between 0 and 255
corresponding to an underlying interface of the target node, NDSCPs
encodes an integer value between 1 and 64 indicating the number of
Differentiated Services Code Point (DSCP) octets that follow. Each
DSCP octet is a 6-bit integer DSCP value followed by a 2-bit
Preference ("Prf") value. Each DSCP value encodes an integer between
0 and 63 associated with this Link ID, where the value 0 means
"default" and other values are interpreted as specified in [RFC2474].
The 'Prf' qualifier for each DSCP value is set to the value 0
("deprecated'), 1 ("low"), 2 ("medium"), or 3 ("high") to indicate a
preference level for packet forwarding purposes. 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 the
encapsulation IP address family is IPv4, IP Address is formed as an
IPv4-mapped IPv6 address [RFC4291].

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 Link Registration

When an administrative authority first deploys a set of AERO Relays
and Servers that comprise an AERO link, they also assign a unique
domain name for the link, e.g., "linkupnetworks.example.com". Next,
if administrative policy permits Clients within the domain to serve
as correspondent nodes for Internet mobile nodes, the administrative
authority adds a Fully Qualified Domain Name (FQDN) for each of the
AERO link's ASPs to the Domain Name System (DNS) [RFC1035]. The FQDN
is based on the suffix "aero.linkupnetworks.net" with a prefix formed
from the wildcard-terminated reverse mapping of the ASP

[RFC3596][RFC4592], and resolves to a DNS PTR resource record. For
example, for the ASP '2001:db8:1::/48' within the domain name
"linkupnetworks.example.com", the DNS database contains:

'*.1.0.0.0.8.b.d.0.1.0.0.2.aero.linkupnetworks.net. PTR
linkupnetworks.example.com'

This DNS registration advertises the AERO link's ASPs to prospective
correspondent nodes.

3.6. AERO Interface Initialization

3.6.1. AERO Relay Behavior

When a Relay enables an AERO interface, it first assigns an
administratively provisioned link-local address fe80::ID to the
interface. Each fe80::ID address MUST be unique among all AERO nodes
on the link, and MUST NOT collide with any potential AERO addresses
nor the special addresses fe80:: and fe80::ffff:ffff:ffff:ffff. (The
fe80::ID addresses are typically taken from the available range
fe80::/96, e.g., as fe80::1, fe80::2, fe80::3, etc.) The Relay then
engages in a dynamic routing protocol session with all Servers on the
link (see: Section 3.7), and advertises its assigned ASP prefixes
into the native IP Internetwork.

Each Relay subsequently maintains an IP forwarding table entry for
each Client-Server association, and maintains a neighbor cache entry
for each Server on the link. Relays exchange NS/NA messages with
AERO link neighbors the same as for any AERO node, however they
typically do not perform explicit Neighbor Unreachability Detection
(NUD) (see: Section 3.18) since the dynamic routing protocol already
provides reachability confirmation.

3.6.2. AERO Server Behavior

When a Server enables an AERO interface, it assigns an
administratively provisioned link-local address fe80::ID the same as
for Relays. The Server further configures a DHCPv6 server function
to facilitate DHCPv6 PD exchanges with AERO Clients. The Server
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 control message exchanges. Each Server also engages
in a dynamic routing protocol with each Relay on the link (see:
Section 3.7).

When the Server receives an NS/RS message from a Client on the AERO
interface it returns an NA/RA message but does not update the
neighbor cache. The Server further provides a simple conduit between

AERO interface neighbors. 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.

3.6.3. AERO Client Behavior

When a Client enables an AERO interface, it uses the special address fe80::ffff:ffff:ffff:ffff to obtain an ACP from an AERO Server via DHCPv6 PD. Next, it assigns the corresponding AERO address to the AERO interface and creates a neighbor cache entry for the Server, i.e., the PD exchange bootstraps autoconfiguration of a unique link-local address. The Client maintains a neighbor cache entry for each of its Servers and each of its active correspondent 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 correspondent Clients (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 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 its Servers or correspondent Clients. Instead, it can set a single "route-to-interface" default route in the IP forwarding table, and all forwarding decisions can be made within the AERO interface based on neighbor cache entries. (On systems in which adding a default route would violate security policy, the default route could instead be installed via a "synthesized RA", e.g., as discussed in Section 3.15.2.)

3.6.4. AERO Forwarding Agent Behavior

When a Forwarding Agent enables an AERO interface, it assigns the same link-local address(es) as the companion AERO Server. The Forwarding Agent thereafter provides data plane forwarding services based solely on the forwarding information assigned to it by the companion AERO Server.

3.7. AERO Link Routing System

Relays require full topology knowledge of all ACP/Server associations for the ASPs they service, while individual Servers at a minimum only need to know the ACPs for 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) using an AS Number (ASN) that is unique within the BGP instance,
and each Server further peers with each Relay but does not peer with
other Servers. Similarly, Relays do not peer with each other, since
they will reliably 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 ACPs, and
dynamically announces new ACPs and withdraws departed ACPs in its BGP
updates to Relays. Clients are expected to remain associated with
their current Servers for extended timeframes, however Servers SHOULD
selectively suppress BGP updates for impatient Clients that
repeatedly associate and disassociate with them in order to dampen
routing churn.

Each Relay configures a black-hole route for each ASP associated with
the AERO link. By black-holing the ASPs, the Relay will maintain
active forwarding table entries only for the ACPs that are currently
active, and all other ACPs will correctly result in destination
unreachable failures due to the black hole route.

Scaling properties of the AERO routing system are limited by the
number of BGP routes that can be carried by Relays. Assuming O(10^6)
as a reasonable maximum number of BGP routes, this means that O(10^6)
Clients can be serviced by a single set of Relays. A means of
increasing scaling would be to assign a different set of Relays for
each set of ASPs. In that case, each Server still peers with each
Relay, but the Server institutes route filters so that each set of
Relays only receives BGP updates for the ACPs they aggregate. For
example, if the ASP for the AERO link is 2001:db8::/32, a first set
of Relays could service the ASP segment 2001:db8::/40, a second set
of Relays could service 2001:db8:0100::/40, a third set could service
2001:db8:0200::/40, etc.

Assuming up to O(10^3) sets of Relays, the system can then
accommodate O(10^9) Clients with no additional overhead for Servers
and Relays. In this way, each set of Relays services a specific set
of ASPs that they advertise to the native routing system outside of
the AERO link, and each Server configures ASP-specific routes that
list the correct set of Relays as next hops. This arrangement also
allows for natural incremental deployment, and can support small
scale initial deployments followed by dynamic deployment of

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

3.8. 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. Each entry
maintains the mapping between the neighbor's fe80::ID network-layer
address and corresponding link-layer address.

Static neighbor cache entries are created though DHCPv6 PD exchanges
and remain in place for durations bounded by prefix lifetimes. AERO
Servers maintain static neighbor cache entries for the ACPs of each
of their associated Clients, and AERO Clients maintain a static
neighbor cache entry for each of their associated Servers. When an
AERO Server sends a DHCPv6 Reply message response to a Client's
DHCPv6 Solicit/Request, Rebind or Renew message, it creates or
updates a static neighbor cache entry based on the AERO address
corresponding to the Client's ACP 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 or updated based on
receipt of an IPv6 ND message, and are garbage-collected if not used
within a bounded timescale. AERO Clients maintain dynamic neighbor
cache entries for each of their active correspondent Client ACPs 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 in the neighbor cache entry to ACCEPT_TIME
seconds and uses this value to determine whether packets received

from the correspondent 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 in the neighbor cache entry to FORWARD_TIME seconds and uses
this value to determine whether packets can be sent directly to the
correspondent. The Client also sets a "MaxRetry" variable to
MAX_RETRY to limit the number of keepalives sent when a correspondent
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 correspondent'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.

3.9. 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 may
have 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 sub-IP layer forwarding agents. 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 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 a permanent neighbor cache entry for a
Relay as the link-layer address for encapsulation.

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 a permanent neighbor
cache entry for a 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 does not match 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 a link-
local address or a non-link-local address that matches an ASP, and
there is a more-specific ACP entry in the IP forwarding table, the
Relay uses the link-layer address in the corresponding neighbor cache
entry as the link-layer address for encapsulation and re-admits the
packet into the AERO link. When an IP packet enters a Relay's AERO
interface from either the network layer or link-layer, and the
packet's destination address matches an ASP but there is no more-

specific ACP entry, the Relay drops the packet and returns an ICMP
Destination Unreachable message (see: Section 3.14).

When an AERO Server receives a packet from a Relay via the AERO
interface, the Server MUST NOT forward the packet back to the same or
a different Relay.

When an AERO Relay receives a packet from a Server via the AERO
interface, the Relay MUST NOT forward the packet back to the same
Server.

When an AERO node re-admits a packet into the AERO link without
involving the network layer, the node MUST NOT decrement the network
layer TTL/Hop-count.

When an AERO node forwards a data packet to the primary link-layer
address of a Server, it may receive Redirect messages with an SLLAO
that include the link-layer address of an AERO Forwarding Agent. The
AERO node SHOULD record the link-layer address in the neighbor cache
entry for the neighbor and send subsequent data packets via this
address instead of the Server's primary address (see: Section 3.16).

3.10. AERO Interface Encapsulation and Re-encapsulation

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

The AERO interface encapsulates packets per the Generic UDP
Encapsulation (GUE) encapsulation procedures in
[I-D.ietf-nvo3-gue][I-D.herbert-gue-fragmentation]. During
encapsulation, the AERO interface copies the "TTL/Hop Limit", "Type
of Service/Traffic Class" [RFC2983], "Flow Label"[RFC6438].(for IPv6)
and "Congestion Experienced" [RFC3168] values in the packet's IP
header into the corresponding fields in the encapsulation IP header.
For packets undergoing re-encapsulation, the AERO interface instead
copies the "TTL/Hop Limit", "Type of Service/Traffic Class", "Flow
Label" and "Congestion Experienced" values in the original
encapsulation IP header into the corresponding fields in the new
encapsulation IP header, i.e., the values are transferred between
encapsulation headers and *not* copied from the encapsulated packet's
network-layer header.

The AERO interface next sets the UDP source port to a constant value
that it will use in each successive packet it sends, and sets the UDP
length field to the length of the encapsulated packet plus 8 bytes
for the UDP header itself, plus the length of the GUE header. 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. For packets sent to a correspondent Client, the AERO interface
sets the UDP destination port to the port value stored in the
neighbor cache entry for this correspondent. The AERO interface also
sets the UDP checksum field per the procedures specified in
[I-D.ietf-nvo3-gue].

The AERO interface next sets the IP protocol number in the
encapsulation header to 17 (i.e., the IP protocol number for UDP).
When IPv4 is used as the encapsulation protocol, the AERO interface
sets the DF bit as discussed in Section 3.13.

3.11. AERO Interface Decapsulation

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. Decapsulation is per the
procedures specified in [I-D.ietf-nvo3-gue].

3.12. AERO Interface Data Origin Authentication

AERO nodes employ simple data origin authentication procedures for
encapsulated packets they receive from other nodes on the AERO link.
In particular:

o AERO Relays and Servers accept encapsulated packets with a link-
   layer source address that matches a permanent neighbor cache
   entry.

o AERO Servers accept authentic encapsulated DHCPv6 messages from
   Clients, and create or update a static neighbor cache entry for
   the source based on the specific message type.

o AERO Servers accept encapsulated packets if there is a neighbor
   cache entry with an AERO address that matches the packet's
   network-layer source address and with a link-layer address that
   matches the packet's link-layer source address.

o AERO Clients accept encapsulated packets if there is a static
   neighbor cache entry with a link-layer source address that matches
   the packet's link-layer source address.

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

Note that this simple data origin authentication is effective in
environments in which link-layer addresses cannot be spoofed. In
other environments, each AERO message must include a signature that
the recipient can use to authenticate the message origin.

3.13. AERO Interface MTU and Fragmentation

The AERO interface is the node's point of attachment to the AERO
link. AERO links over IP networks have a maximum link MTU of 64KB
minus the encapsulation overhead (i.e., 64KB-ENCAPS), since the
maximum packet size in the base IP specifications is 64KB
[RFC0791][RFC2460]. While IPv6 jumbograms can be up to 4GB
[RFC2675], they are considered optional for IPv6 nodes [RFC6434] and
therefore out of scope for this document.

AERO interfaces are considered to have an indefinite MTU , i.e.,
instead of clamping the MTU to a finite size. The MTU for each
destination AERO interface is therefore constrained by the minimum of
64KB, the MTU of the underlying interface used for tunneling, and the
path MTU within the tunnel (minus ENCAPS in each case).

IPv6 specifies a minimum link MTU of 1280 bytes [RFC2460]. This is
the minimum packet size the AERO interface MUST admit without
returning an ICMP Packet Too Big (PTB) message. Although IPv4
specifies a smaller minimum link MTU of 68 bytes [RFC0791], AERO
interfaces also observe a 1280 byte minimum for IPv4.

The vast majority of links in the Internet configure an MTU of at
least 1500 bytes. Original source hosts have therefore become
conditioned to expect that IP packets up to 1500 bytes in length will
either be delivered to the final destination or a suitable PTB
message returned. However, PTB messages may be lost in the network
[RFC2923] resulting in failure of the IP Path MTU Discovery (PMTUD)
mechanisms [RFC1191][RFC1981], or crafted for malicious purposes such
as denial of service. For these reasons, the source AERO interface
(i.e., the tunnel ingress) sends encapsulated packets to the
destination AERO interface (i.e., the tunnel egress) subject to
whether standard PMTUD can be leveraged within the specific
deployment model. The two cases for consideration are as follows:

3.13.1. All Elements in Same Administrative Domain

When the original source, ingress and egress are all within the same
well-managed administrative domain, the ingress admits a packet into
the tunnel if it is not too large to traverse the tunnel in one piece
or if the path MTU within the tunnel is not yet known. Otherwise,
the ingress drops the packet and sends a PTB message back to the
original source. Additionally, the ingress SHOULD translate any PTB

   messages it receives from a router on the path to the egress and
   cache the MTU value as a maximum size for large packet admission.

   These procedures apply when the MTU for this egress is no smaller
   than (1280+ENCAPS) bytes. Otherwise, the ingress can either shut
   down the tunnel or begin fragmenting packets that are no larger than
   1280 bytes but larger than the MTU minus ENCAPS using tunnel
   fragmentation as specified in Section 3.13.2. This exactly parallels
   the standard behavior specified in [RFC2473] except that, when the
   original packet is an IPv4 packet with DF=0, the ingress uses IPv4
   fragmentation to fragment the original packet when necessary before
   encapsulation as specified in Section 3.13.2.

3.13.2. Some Elements in Different Administrative Domain

   When the original source, ingress and egress are not all within the
   same well-managed administrative domain, the ingress admits all
   packets up to 1500 bytes in length even if some fragmentation is
   necessary, and admits larger packets without fragmentation in case
   they are able to traverse the tunnel in one piece. In that case, the
   ingress SHOULD process any PTB messages it receives with an MTU size
   larger than 1500 bytes as a packet size limitation the same as
   described in Section 3.13.1.

   Several factors must be considered when fragmentation is needed. For
   AERO links over IPv4, the IP ID field is only 16 bits in length,
   meaning that fragmentation at high data rates could result in data
   corruption due to reassembly misassociations [RFC6864][RFC4963] (see:
   Section 3.13.4). For AERO links over both IPv4 and IPv6, studies
   have also shown that IP fragments are dropped unconditionally over
   some network paths [I-D.taylor-v6ops-fragdrop]. For these reasons,
   when fragmentation is needed the ingress inserts a GUE fragment
   header [I-D.herbert-gue-fragmentation] and applies tunnel
   fragmentation as described in Section 3.1.7 of [RFC2764]. Since the
   fragment header reduces the room available for packet data, but the
   original source has no way to control its insertion, the ingress MUST
   include the fragment header length in the ENCAPS length even for
   packets in which the header does not appear.

   The ingress therefore sends encapsulated packets to the egress
   according to the following algorithm:

   o For IP packets that are no larger than (1280-ENCAPS) bytes, the
      ingress encapsulates the packet and admits it into the tunnel
      without fragmentation. For IPv4 AERO links, the ingress sets the
      Don't Fragment (DF) bit to 0 so that these packets will be
      delivered to the egress even if there is a restricting link in the
      path, i.e., unless lost due to congestion or routing errors.

o For IP packets that are larger than (1280-ENCAPS) bytes but no
   larger than 1500 bytes, the ingress encapsulates the packet and
   inserts a GUE fragment header. Next, the ingress fragments the
   packet into two non-overlapping fragments where the first fragment
   (including ENCAPS) is no larger than 1024 bytes and the second is
   no larger than the first. Each fragment consists of identical
   IP/UDP/GUE encapsulation headers followed by the fragment of the
   encapsulated packet itself. The ingress then admits both
   fragments into the tunnel, and for IPv4 sets the DF bit to 0 in
   the IP encapsulation header. These fragmented encapsulated
   packets will be delivered to the egress, which reassembles them
   into a whole packet. The egress therefore MUST be capable of
   reassembling packets up to 1500+ENCAPS bytes in length; hence, it
   is RECOMMENDED that the egress be capable of reassembling at least
   2KB.

o For IPv4 packets that are larger than 1500 bytes and with the DF
   bit set to 0, the ingress uses ordinary IPv4 fragmentation to
   break the unencapsulated packet into a minimum number of non-
   overlapping fragments where the first fragment is no larger than
   1024-ENCAPS and all other fragments are no larger than the first
   fragment. The ingress then encapsulates each fragment (and for
   IPv4 sets the DF bit to 0) then admits them into the tunnel.
   These fragments will be delivered to the final destination via the
   egress.

o For all other IP packets, if the packet is too large to enter the
   underlying interface following encapsulation, the ingress drops
   the packet and returns a network-layer (L3) PTB message to the
   original source with MTU set to the larger of 1500 bytes or the
   underlying interface MTU minus ENCAPS. Otherwise, the ingress
   encapsulates the packet and admits it into the tunnel without
   fragmentation (and for IPv4 sets the DF bit to 1) and translates
   any link-layer (L2) PTB messages it may receive from the network
   with an MTU size larger than 1500 bytes into corresponding L3 PTB
   messages to send to the original source as specified in
   Section 3.14. Since both L2 and L3 PTB messages may either be
   lost or contain insufficient information, however, it is
   RECOMMENDED that original sources that send unfragmentable IP
   packets larger than 1500 bytes use Packetization Layer Path MTU
   Discovery (PLPMTUD) [RFC4821].

While sending packets according to the above algorithm, the ingress
MAY also send 1500 byte or larger probe packets to determine whether
they can reach the egress without fragmentation. If the probes
succeed, the ingress can discontinue fragmentation and (for IPv4) set
DF to 1. Since the path MTU within the tunnel may fluctuate due to
routing changes, the ingress SHOULD continue to send additional

probes subject to rate limiting and SHOULD process any L2 PTB
messages as an indication that the path MTU may have decreased. If
the path MTU within the tunnel becomes insufficient, the source MUST
resume fragmentation.

To construct a probe, the ingress can prepare an NS message with a
Nonce option plus trailing NULL padding octets added to the probe
length without including the length of the padding in the IPv6
Payload Length field, but with the length included in the
encapsulating IP header. The ingress then encapsulates the padded NS
message in the encapsulation headers (and for IPv4 sets DF to 1) then
sends the message to the egress. If the egress returns a solicited
NA message with a matching Nonce option, the ingress deems the probe
successful. (Note that other means of probing such as sending a
self-addressed data packet that will be looped back if received by
the egress are also permissible.)

When probing is used, it is essential that probes follow the same
paths used to convey actual data packets to avoid path variations due
to Equal Cost MultiPath (ECMP) and Link Aggregation Gateway (LAG)
equipment in the path. In order to ensure that probes follow the
same paths as data packets, the ingress sets both the Differentiated
Service Code Point (DSCP) [RFC2983] and (for IPv6) Flow Label
[RFC6438] fields to a constant value during encapsulation instead of
copying these values from the original packet. In that case, it is
RECOMMENDED that the ingress set these fields to the value 0.

Note that an exception to the above rules occurs when the original
source and ingress are both with in the same well-managed
administrative domain. In that case, if the underlying interface
used by the ingress for tunneling configures an MTU smaller than
(1500+HLEN) bytes, the ingress MAY drop packets that are larger than
1280 bytes and larger than the underlying interface MTU following
encapsulation, and return a PTB message to the original source.

3.13.3. Accommodating Large Control Messages

Control messages (i.e., IPv6 ND, DHCPv6, etc.) MUST be accommodated
even if some fragmentation is necessary. These packets are therefore
accommodated through a modification of the second rule in the above
algorithm as follows:

o For control messages that are larger than (1280-ENCAPS) bytes, the
   ingress encapsulates the packet and inserts a GUE fragment header.
   Next, the ingress uses GUE fragmentation
   [I-D.herbert-gue-fragmentation] to break the packet into a minimum
   number of non-overlapping fragments where the first fragment
   (including ENCAPS) is no larger than 1024 bytes and the remaining

fragments are no larger than the first. The ingress then
encapsulates each fragment (and for IPv4 sets the DF bit to 0)
then admits them into the tunnel.

Control messages that exceed the 2KB minimum reassembly size rarely
occur in the modern era, however the egress SHOULD be able to
reassemble them if they do. This means that the egress SHOULD
include a configuration knob allowing the operator to set a larger
reassembly buffer size if large control messages become more common
in the future.

The ingress MAY send large control messages without fragmentation if
there is assurance that large packets can traverse the tunnel without
fragmentation.

The ingress MAY send 1500 byte or larger probe packets as specified
in the previous section to determine a size for which fragmentation
can be avoided.

3.13.4. Integrity

When fragmentation is needed, there must be assurance that reassembly
can be safely conducted without incurring data corruption. Sources
of corruption can include implementation errors, memory errors and
misassociations of fragments from a first datagram with fragments of
another datagram. The first two conditions (implementation and
memory errors) are mitigated by modern systems and implementations
that have demonstrated integrity through decades of operational
practice. The third condition (reassembly misassociations) must be
accounted for by AERO.

The fragmentation procedure described in the above algorithms can
reuse standard IPv6 fragmentation and reassembly code. Since the GUE
fragment header includes a 32-bit ID field, there would need to be
2^{32} packets alive in the network before a second packet with a
duplicate ID enters the system with the (remote) possibility for a
reassembly misassociation. For 1280 byte packets, and for a maximum
network lifetime value of 60 seconds[RFC2460], this means that the
ingress would need to produce ~$(7 * 10^{12})$ bits/sec in order for a
duplication event to be possible. This exceeds the bandwidth of data
link technologies of the modern era, but not necessarily so going
forward into the future. Although wireless data links commonly used
by AERO Clients support vastly lower data rates, the aggregate data
rates between AERO Servers and Relays may be substantial. However,
high speed data links in the network core are expected to configure
larger MTUs, e.g., 4KB, 8KB or even larger such that unfragmented
packets can be used. Hence, no integrity check is included to cover
fragmentation and reassembly procedures.

When the ingress sends an IPv4-encapsulated packet with the DF bit
set to 0 in the above algorithms, there is a chance that the packet
may be fragmented by an IPv4 router somewhere within the tunnel.
Since the largest such packet is only 1280 bytes, however, it is very
likely that the packet will traverse the tunnel without incurring a
restricting link. Even when a link within the tunnel configures an
MTU smaller than 1280 bytes, it is very likely that it does so due to
limited performance characteristics [RFC3819]. This means that the
tunnel would not be able to convey fragmented IPv4-encapsulated
packets fast enough to produce reassembly misassociations, as
discussed above. However, AERO must also account for the possibility
of tunnel paths that include "poorly managed" IPv4 link MTUs due to
misconfigurations.

Since the IPv4 header includes only a 16-bit ID field, there would
only need to be 2^16 packets alive in the network before a second
packet with a duplicate ID enters the system. For 1280 byte packets,
and for a maximum network lifetime value of 120 seconds[RFC0791],
this means that the ingress would only need to produce ~(5 *10^6)
bits/sec in order for a duplication event to be possible - a value
that is well within range for many modern wired and wireless data
link technologies.

Therefore, if there is strong operational assurance that no IPv4
links capable of supporting data rates of 5Mbps or more configure an
MTU smaller than 1280 the ingress MAY omit an integrity check for the
IPv4 fragmentation and reassembly procedures; otherwise, the ingress
SHOULD include an integrity check. When an upper layer encapsulation
(e.g., IPsec) already includes an integrity check, the ingress need
not include an additional check. Otherwise, the ingress calculates
the UDP checksum over the encapsulated packet and writes the value
into the UDP encapsulation header, i.e., instead of writing the value
0. The egress will then verify the UDP checksum and discard the
packet if the checksum is incorrect.

3.14. AERO Interface Error Handling

When an AERO node admits encapsulated packets into the AERO
interface, it may receive link-layer (L2) or network-layer (L3) error
indications.

An L2 error indication is an ICMP error message generated by a router
on the path to the neighbor or by the neighbor itself. The message
includes an IP header with the address of the node that generated the
error as the source address and with the link-layer address of the
AERO node as the destination address.

The IP header is followed by an ICMP header that includes an error
Type, Code and Checksum. For ICMPv6 [RFC4443], the error Types
include "Destination Unreachable", "Packet Too Big (PTB)", "Time
Exceeded" and "Parameter Problem". For ICMPv4 [RFC0792], the error
Types include "Destination Unreachable", "Fragmentation Needed" (a
Destination Unreachable Code that is analogous to the ICMPv6 PTB),
"Time Exceeded" and "Parameter Problem".

The ICMP header is followed by the leading portion of the packet that
generated the error, also known as the "packet-in-error". For
ICMPv6, [RFC4443] specifies that the packet-in-error includes: "As
much of invoking packet as possible without the ICMPv6 packet
exceeding the minimum IPv6 MTU" (i.e., no more than 1280 bytes). For
ICMPv4, [RFC0792] specifies that the packet-in-error includes:
"Internet Header + 64 bits of Original Data Datagram", however
[RFC1812] Section 4.3.2.3 updates this specification by stating: "the
ICMP datagram SHOULD contain as much of the original datagram as
possible without the length of the ICMP datagram exceeding 576
bytes".

The L2 error message format is shown in Figure 3:

```
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     ~                               ~
     |         L2 IP Header of       |
     |          error message        |
     ~                               ~
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |         L2 ICMP Header        |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
     ~                               ~  P
     |    IP and other encapsulation |  a
     | headers of original L3 packet |  c
     ~                               ~  k
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  e
     ~                               ~  t
     |          IP header of         |
     |       original L3 packet      |  i
     ~                               ~  n
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     ~                               ~  e
     |      Upper layer headers and  |  r
     |       leading portion of body |  r
     |    of the original L3 packet  |  o
     ~                               ~  r
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
```
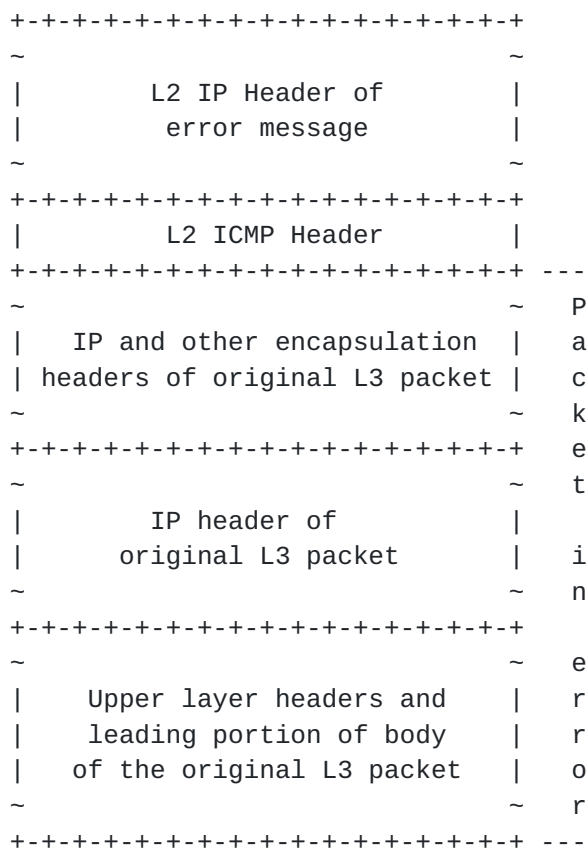
Figure 3: AERO Interface L2 Error Message Format

The AERO node rules for processing these L2 error messages is as
follows:

o When an AERO node receives an L2 Parameter Problem message, it
   processes the message the same as described as for ordinary ICMP
   errors in the normative references [RFC0792][RFC4443].

o When an AERO node receives persistent L2 IPv4 Time Exceeded
   messages, the IP ID field may be wrapping before earlier fragments
   have been processed. In that case, the node SHOULD begin
   including IPv4 integrity checks (see: Section 3.13.4).

o When an AERO Client receives persistent L2 Destination Unreachable
   messages in response to tunneled packets that it sends to one of
   its dynamic neighbor correspondents, the Client SHOULD test the
   path to the correspondent using Neighbor Unreachability Detection
   (NUD) (see Section 3.18). If NUD fails, the Client SHOULD set
   ForwardTime for the corresponding dynamic neighbor cache entry to
   0 and allow future packets destined to the correspondent to flow
   through a Server.

o When an AERO Client receives persistent L2 Destination Unreachable
   messages in response to tunneled packets that it sends to one of
   its static neighbor Servers, the Client SHOULD test the path to
   the Server using NUD. If NUD fails, the Client SHOULD delete the
   neighbor cache entry and attempt to associate with a new Server.

o When an AERO Server receives persistent L2 Destination Unreachable
   messages in response to tunneled packets that it sends to one of
   its static neighbor Clients, the Server SHOULD test the path to
   the Client using NUD. If NUD fails, the Server SHOULD cancel the
   DHCPv6 PD for the Client's ACP, withdraw its route for the ACP
   from the AERO routing system and delete the neighbor cache entry
   (see Section 3.18 and Section 3.19).

o When an AERO Relay or Server receives an L2 Destination
   Unreachable message in response to a tunneled packet that it sends
   to one of its permanent neighbors, it discards the message since
   the routing system is likely in a temporary transitional state
   that will soon re-converge.

o When an AERO node receives an L2 PTB message, it translates the
   message into an L3 PTB message if possible (*) and forwards the
   message toward the original source as described below.

To translate an L2 PTB message to an L3 PTB message, the AERO node
first caches the MTU field value of the L2 ICMP header. The node
next discards the L2 IP and ICMP headers, and also discards the

encapsulation headers of the original L3 packet. Next the node
encapsulates the included segment of the original L3 packet in an L3
IP and ICMP header, and sets the ICMP header Type and Code values to
appropriate values for the L3 IP protocol. In the process, the node
writes the maximum of 1500 bytes and (L2 MTU - ENCAPS) into the MTU
field of the L3 ICMP header.

The node next writes the IP source address of the original L3 packet
as the destination address of the L3 PTB message and determines the
next hop to the destination. If the next hop is reached via the AERO
interface, the node uses the IPv6 address "::" or the IPv4 address
"0.0.0.0" as the IP source address of the L3 PTB message. Otherwise,
the node uses one of its non link-local addresses as the source
address of the L3 PTB message. The node finally calculates the ICMP
checksum over the L3 PTB message and writes the Checksum in the
corresponding field of the L3 ICMP header. The L3 PTB message
therefore is formatted as follows:

```
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      ~                               ~
      |          L3 IP Header of       |
      |           error message        |
      ~                               ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |          L3 ICMP Header         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ---
      ~                               ~   p
      |           IP header of         |   k
      |        original L3 packet      |   t
      ~                               ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   i
      ~                               ~   n
      |     Upper layer headers and    |
      |      leading portion of body   |   e
      |     of the original L3 packet  |   r
      ~                               ~   r
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
```

                Figure 4: AERO Interface L3 Error Message Format

After the node has prepared the L3 PTB message, it either forwards
the message via a link outside of the AERO interface without
encapsulation, or encapsulates and forwards the message to the next
hop via the AERO interface.

When an AERO Relay receives an L3 packet for which the destination
address is covered by an ASP, if there is no more-specific routing
information for the destination the Relay drops the packet and

returns an L3 Destination Unreachable message. The Relay first
writes the IP source address of the original L3 packet as the
destination address of the L3 Destination Unreachable message and
determines the next hop to the destination. If the next hop is
reached via the AERO interface, the Relay uses the IPv6 address "::"
or the IPv4 address "0.0.0.0" as the IP source address of the L3
Destination Unreachable message and forwards the message to the next
hop within the AERO interface. Otherwise, the Relay uses one of its
non link-local addresses as the source address of the L3 Destination
Unreachable message and forwards the message via a link outside the
AERO interface.

When an AERO node receives any L3 error message via the AERO
interface, it examines the destination address in the L3 IP header of
the message. If the next hop toward the destination address of the
error message is via the AERO interface, the node re-encapsulates and
forwards the message to the next hop within the AERO interface.
Otherwise, if the source address in the L3 IP header of the message
is the IPv6 address "::" or the IPv4 address "0.0.0.0", the node
writes one of its non link-local addresses as the source address of
the L3 message and recalculates the IP and/or ICMP checksums. The
node finally forwards the message via a link outside of the AERO
interface.

(*) Note that in some instances the packet-in-error field of an L2
PTB message may not include enough information for translation to an
L3 PTB message. In that case, the AERO interface simply discards the
L2 PTB message. It can therefore be said that translation of L2 PTB
messages to L3 PTB messages can provide a useful optimization when
possible, but is not critical for sources that correctly use PLPMTUD.

3.15. AERO Router Discovery, Prefix Delegation and Address Configuration

3.15.1. AERO DHCPv6 Service Model

Each AERO Server configures a DHCPv6 server function to facilitate PD
requests from Clients. Each Server is provisioned with a database of
ACP-to-Client ID mappings for all Clients enrolled in the AERO
system, as well as any information necessary to authenticate each
Client. The Client database is maintained by a central
administrative authority for the AERO link and securely distributed
to all Servers, e.g., via the Lightweight Directory Access Protocol
(LDAP) [RFC4511] or a similar distributed database service.

Therefore, no Server-to-Server DHCPv6 PD delegation state
synchronization is necessary, and Clients can optionally hold
separate delegations for the same ACP from multiple Servers. In this

   way, Clients can associate with multiple Servers, and can receive new
   delegations from new Servers before deprecating delegations received
   from existing Servers.

   AERO Clients and Servers exchange Client link-layer address
   information using an option format similar to the Client Link Layer
   Address Option (CLLAO) defined in [RFC6939]. Due to practical
   limitations of CLLAO, however, AERO interfaces instead use Vendor-
   Specific Information Options as described in the following sections.

3.15.2. 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 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 two-message (i.e., Solicit/Reply) or four-
   message (i.e., Solicit/Advertise/Request/Reply) DHCPv6 PD exchange
   [RFC3315][RFC3633]. The Client's Solicit/Request message includes
   fe80::ffff:ffff:ffff:ffff 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 Solicit/Request message also includes a
   Rapid Commit option, a Client Identifier option with a DHCP Unique
   Identifier (DUID) and an Identity Association for Prefix Delegation
   (IA_PD) option. If the Client is pre-provisioned with an ACP
   associated with the AERO service, it MAY also include the ACP in the
   IA_PD to indicate its preference to the DHCPv6 server.

   The Client also SHOULD include an AERO Link-registration Request
   (ALREQ) option to register one or more links with the Server. The
   Server will include an AERO Link-registration Reply (ALREP) option in
   the corresponding DHCPv6 Reply message as specified in
   Section 3.15.3. (The Client MAY omit the ALREQ option, in which case
   the Server will still include an ALREP option in its Reply with "Link
   ID" set to 0, "DSCP" set to 0, and "Prf" set to 3.)

   The format for the ALREQ option is shown in Figure 5:

```
      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |        OPTION_VENDOR_OPTS        |         option-len (1)        |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                    enterprise-number = 45282                   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  opt-code = OPTION_ALREQ (0)  |          option-len (2)         |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |    Link ID    |  DSCP #1  |Prf|  DSCP #2  |Prf|   ...
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

         Figure 5: AERO Link-registration Request (ALREQ) Option

   In the above format, the Client sets 'option-code' to
   OPTION_VENDOR_OPTS, sets 'option-len (1)' to the length of the option
   following this field, sets 'enterprise-number' to 45282 (see: "IANA
   Considerations"), sets opt-code to the value 0 ("OPTION_ALREQ") and
   sets 'option-len (2)' to the length of the remainder of the option.
   The Client includes appropriate 'Link ID, 'DSCP' and 'Prf' values for
   the underlying interface over which the DHCPv6 PD Solicit/Request
   message will be issued the same as specified for an S/TLLAO
   Section 3.4. The Client MAY include multiple (DSCP, Prf) values with
   this Link ID, with the number of values indicated by option-len (2).
   The Server will register each value with the Link ID in the Client's
   neighbor cache entry. The Client finally includes any necessary
   authentication options to identify itself to the DHCPv6 server, and
   sends the encapsulated DHCPv6 PD Solicit/Request message via the
   underlying interface corresponding to Link ID. (Note that this
   implies that the Client must perform additional Rebind/Reply DHCPv6
   exchanges with the server following the initial PD exchange using
   different underlying interfaces and their corresponding Link IDs if
   it wishes to register additional link-layer addresses and their
   associated DSCPs.)

   When the Client receives its ACP via a DHCPv6 Reply from the AERO
   Server, it creates a static neighbor cache entry with the Server's
   link-local address as the network-layer address and the Server's
   encapsulation address as the link-layer address. The Client then
   considers the link-layer address of the Server as the primary default
   encapsulation address for forwarding packets for which no more-
   specific forwarding information is available. The Client further
   caches any ASPs included in the ALREP option as ASPs to apply to the
   AERO link.

   Next, the Client autoconfigures an AERO address from the delegated
   ACP, assigns the AERO address to the AERO interface and sub-delegates
   the ACP to its attached EUNs and/or the Client's own internal virtual

interfaces. 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 Client can then determine the correct next hops for packets
submitted to the AERO interface by inspecting the neighbor cache.

The Client subsequently renews its ACP delegation through each of its
Servers by performing DHCPv6 Renew/Reply exchanges with the link-
layer address of a Server as the link-layer destination address and
the same options that were used in the initial PD request. Note that
if the Client does not issue a DHCPv6 Renew before the delegation
expires (e.g., if the Client has been out of touch with the Server
for a considerable amount of time) it must re-initiate the DHCPv6 PD
procedure.

Since the Client's AERO address is obtained 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).

When a Client attempts to perform a DHCPv6 PD exchange with a Server
that is too busy to service the request, the Client may receive a
"NoPrefixAvail" status code in the Server's Reply per [RFC3633]. In
that case, the Client SHOULD discontinue DHCPv6 PD attempts through
this Server and try another Server.

3.15.2.1. Autoconfiguration for Constrained Platforms

On some platforms (e.g., popular cell phone operating systems), the
act of assigning a default IPv6 route and/or assigning an address to
an interface may not be permitted from a user application due to
security policy. Typically, those platforms include a TUN/TAP
interface that acts as a point-to-point conduit between user
applications and the AERO interface. In that case, the Client can
instead generate a "synthesized RA" message. The message conforms to
[RFC4861] and is prepared as follows:

o the IPv6 source address is the Client's AERO address

o the IPv6 destination address is all-nodes multicast

o the Router Lifetime is set to a time that is no longer than the
   ACP DHCPv6 lifetime

o the message does not include a Source Link Layer Address Option
   (SLLAO)

   o the message includes a Prefix Information Option (PIO) with a /64
      prefix taken from the ACP as the prefix for autoconfiguration

   The Client then sends the synthesized RA message via the TUN/TAP
   interface, where the operating system kernel will interpret it as
   though it were generated by an actual router. The operating system
   will then install a default route and use StateLess Address
   AutoConfiguration (SLAAC) to configure an IPv6 address on the TUN/TAP
   interface. Methods for similarly installing an IPv4 default route
   and IPv4 address on the TUN/TAP interface are based on synthesized
   DHCPv4 messages [RFC2131].

3.15.2.2. Client DHCPv6 Message Source Address

   In the initial DHCPv6 PD message exchanges, AERO Clients use the
   special IPv6 source address 'fe80::ffff:ffff:ffff:ffff' since their
   AERO addresses are not yet configured. After AERO address
   autoconfiguration, however, AERO Clients can either continue to use
   'fe80::ffff:ffff:ffff:ffff' as the source address for further DHCPv6
   messaging or begin using their AERO address as the source address.

3.15.3. 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/Request on its AERO interface, and the Server is too busy to
   service the message, it returns a Reply with status code
   "NoPrefixAvail" per [RFC3633]. Otherwise, the Server authenticates
   the message.

   If authentication succeeds, the Server determines the correct ACP to
   delegate to the Client by searching the Client database. In
   environments where spoofing is not considered a threat, the Server
   MAY use the Client's DUID as the identification value. Otherwise,
   the Server SHOULD use a signed certificate provided by the Client.

   When the Server delegates the ACP, it also creates an IP forwarding
   table entry so that the AERO routing system will propagate the ACP to
   all Relays that aggregate the corresponding ASP (see: Section 3.7).
   Next, the Server prepares a DHCPv6 Reply message to send to the
   Client while using fe80::ID as the IPv6 source address, the link-
   local address taken from the Client's Solicit/Request as the IPv6
   destination address, the Server's link-layer address as the source
   link-layer address, and the Client's link-layer address as the

destination link-layer address. The server also includes an IA_PD
option with the delegated ACP. Since the Client may experience a
fault that prevents it from issuing a DHCPv6 Release before departing
from the network, Servers should set a short prefix lifetime (e.g.,
40 seconds) so that stale prefix delegation state can be flushed out
of the network.

The Server also includes an ALREP option that includes the UDP Port
Number and IP Address values it observed when it received the ALREQ
in the Client's original DHCPv6 message (if present) followed by the
ASP(s) for the AERO link. The ALREP option is formatted as shown in
Figure 6:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       OPTION_VENDOR_OPTS       |          option-len (1)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   enterprise-number = 45282                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  opt-code = OPTION_ALREP (1)  |          option-len (2)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Link ID    |    Reserved    |        UDP Port Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                          IP Address                           +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+           AERO Service Prefix (ASP) #1     +-+-+-+-+-+-+-+-+-+
|                                            | Prefix Len    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+           AERO Service Prefix (ASP) #2     +-+-+-+-+-+-+-+-+-+
|                                            | Prefix Len    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
~                                                               ~
```

Figure 6: AERO Link-registration Reply (ALREP) Option

In the ALREP, the Server sets 'option-code' to OPTION_VENDOR_OPTS,
sets 'option-length (1)' to the length of the option, sets

'enterprise-number' to 45282 (see: "IANA Considerations"), sets opt-
code to OPTION_ALREP (1), and sets 'option-len (2)' to the length of
the remainder of the option. Next, the Server sets 'Link ID' to the
same value that appeared in the ALREQ, sets Reserved to 0 and sets
'UDP Port Number' and 'IP address' to the Client's link-layer
address. The Server next includes one or more ASP with the IP prefix
as it would appear in the interface identifier portion of the
corresponding AERO address (see: Section 3.3), except that the low-
order 8 bits of the ASP field encode the prefix length instead of the
low-order 8 bits of the prefix. The longest prefix that can
therefore appear as an ASP is /56 for IPv6 or /24 for IPv4. (Note
that if the Client did not include an ALREQ option in its DHCPv6
message, the Server MUST still include an ALREP option in the
corresponding reply with 'Link ID' set to 0.)

When the Server admits the DHCPv6 Reply message into the AERO
interface, it creates a static neighbor cache entry for the Client's
AERO address with lifetime set to no more than the delegation
lifetime and the Client's link-layer address as the link-layer
address for the Link ID specified in the ALREQ. The Server then uses
the Client link-layer address information in the ALREQ option as the
link-layer address for encapsulation based on the (DSCP, Prf)
information.

After the initial DHCPv6 PD exchange, the AERO Server maintains the
neighbor cache entry for the Client until the delegation lifetime
expires. If the Client issues a Renew/Reply exchange, the Server
extends the lifetime. If the Client issues a Release/Reply, or if
the Client does not issue a Renew/Reply before the lifetime expires,
the Server deletes the neighbor cache entry for the Client and
withdraws the IP route from the AERO routing system.

3.15.3.1. Lightweight DHCPv6 Relay Agent (LDRA)

AERO Clients and Servers are always on the same link (i.e., the AERO
link) from the perspective of DHCPv6. However, in some
implementations the DHCPv6 server and AERO interface driver may be
located in separate modules. In that case, the Server's AERO
interface driver module acts as a Lightweight DHCPv6 Relay Agent
(LDRA)[RFC6221] to relay DHCPv6 messages to and from the DHCPv6
server module.

When the LDRA receives a DHCPv6 message from a client, it prepares an
ALREP option the same as described above then wraps the option in a
Relay-Supplied DHCP Option option (RSOO) [RFC6422]. The LDRA then
incorporates the option into the Relay-Forward message and forwards
the message to the DHCPv6 server.

When the DHCPv6 server receives the Relay-Forward message, it caches
the ALREP option and authenticates the encapsulated DHCPv6 message.
The DHCPv6 server subsequently ignores the ALREQ option itself, since
the relay has already included the ALREP option.

When the DHCPv6 server prepares a Reply message, it then includes the
ALREP option in the body of the message along with any other options,
then wraps the message in a Relay-Reply message. The DHCPv6 server
then delivers the Relay-Reply message to the LDRA, which discards the
Relay-Reply wrapper and delivers the DHCPv6 message to the Client.

3.15.4. Deleting Link Registrations

After an AERO Client registers its Link IDs and their associated
(DSCP,Prf) values with the AERO Server, the Client may wish to delete
one or more Link registrations, e.g., if an underlying link becomes
unavailable. To do so, the Client prepares a DHCPv6 Rebind message
that includes an AERO Link-registration Delete (ALDEL) option and
sends the Rebind message to the Server over any available underlying
link. The ALDEL option is formatted as shown in Figure 7:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       OPTION_VENDOR_OPTS       |        option-len (1)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   enterprise-number = 45282                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  opt-code = OPTION_ALDEL (2)  |        option-len (2)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Link ID #1  |  Link ID #2   |  Link ID #3   |    ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```
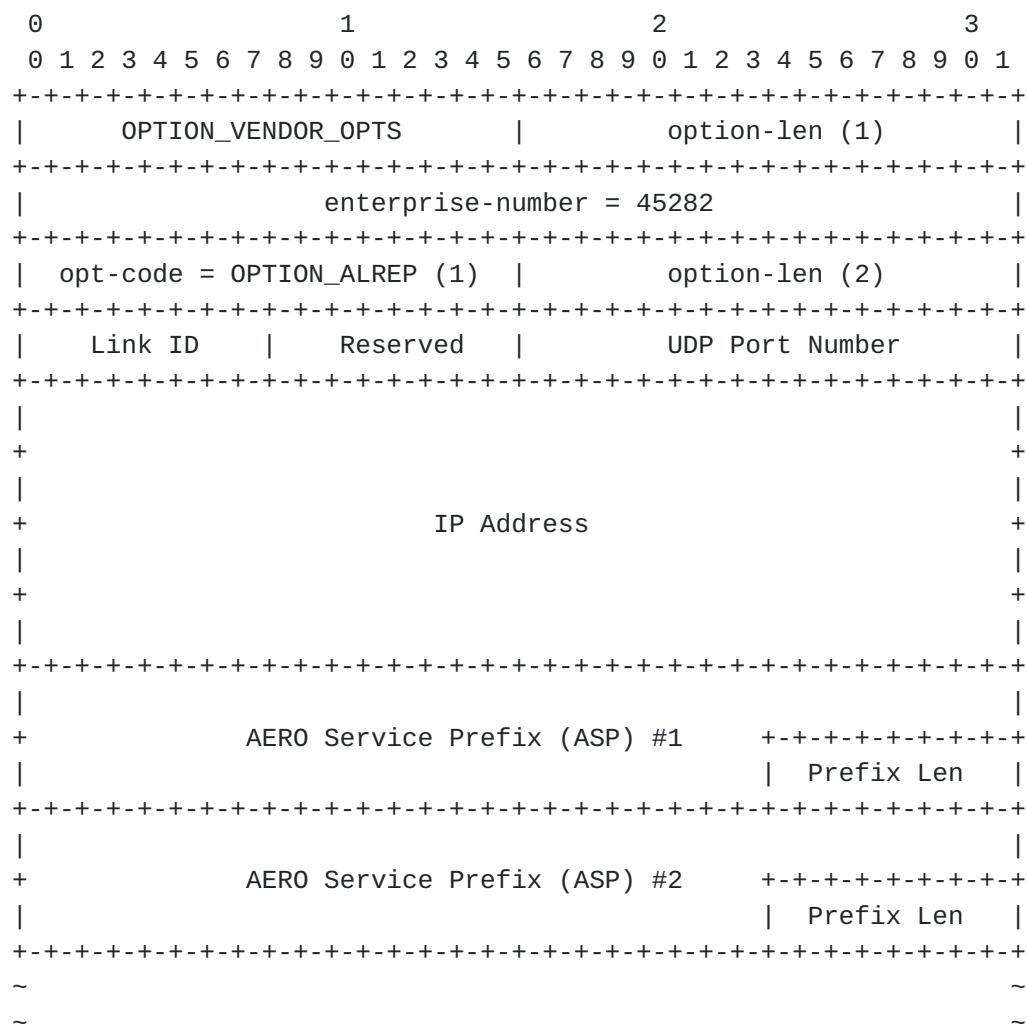
Figure 7: AERO Link-registration Delete (ALDEL) Option

In the ALDEL, the Client sets 'option-code' to OPTION_VENDOR_OPTS,
sets 'option-length (1)' to the length of the option, sets
'enterprise-number' to 45282 (see: "IANA Considerations"), sets
optcode to OPTION_ALDEL (2), and sets 'option-len (2)' to the length
of the remainder of the option. Next, the Server includes each 'Link
ID' value that it wishes to delete.

If the Client wishes to discontinue use of a Server and thereby
delete all of its Link ID associations, it must use a DHCPv6 Release/
Reply exchange to delete the entire neighbor cache entry, i.e.,
instead of using a DHCPv6 Rebind/Reply exchange with one or more
ALDEL options.

3.16. AERO Forwarding Agent Behavior

   AERO Servers MAY associate with one or more companion AERO Forwarding
   Agents as platforms for offloading high-speed data plane traffic.
   When an AERO Server receives a Client's DHCPv6
   Solicit/Request/Renew/Rebind/Release message, it services the message
   then forwards the corresponding Reply message to the Forwarding
   Agent. When the Forwarding Agent receives the Reply message, it
   creates, updates or deletes a neighbor cache entry with the Client's
   AERO address and link-layer information included in the Reply
   message. The Forwarding Agent then forwards the Reply message back
   to the AERO Server, which forwards the message to the Client. In
   this way, Forwarding Agent state is managed in conjunction with
   Server state, with the Client responsible for reliability. If the
   Client subsequently disappears without issuing a Release, the Server
   is responsible for purging stale state by sending synthesized Reply
   messages to the Forwarding Agent.

   When an AERO Server receives a data packet on an AERO interface with
   a network layer destination address for which it has distributed
   forwarding information to a Forwarding Agent, the Server returns a
   Redirect message to the source neighbor (subject to rate limiting)
   then forwards the data packet as usual. The Redirect message
   includes a TLLAO with the link-layer address of the Forwarding
   Engine.

   When the source neighbor receives the Redirect message, it SHOULD
   record the link-layer address in the TLLAO as the encapsulation
   addresses to use for sending subsequent data packets. However, the
   source MUST continue to use the primary link-layer address of the
   Server as the encapsulation address for sending control messages.

3.17. AERO Intradomain Route Optimization

   When a source Client forwards packets to a prospective correspondent
   Client within the same AERO link domain (i.e., one for which the
   packet's destination address is covered by an ASP), the source Client
   initiates an intra-domain AERO route optimization procedure. It is
   important to note that this procedure is initiated by the Client; if
   the procedure were initiated by the Server, the Server would have no
   way of knowing whether the Client was actually able to contact the
   correspondent over the route-optimized path.

   The procedure is based on an exchange of IPv6 ND messages using a
   chain of AERO Servers and Relays as a trust basis. This procedure is
   in contrast to the Return Routability procedure required for route
   optimization to a correspondent Client located in the Internet as

described in [Section 3.22](#). The following sections specify the AERO
intradomain route optimization procedure.

[3.17.1](#). Reference Operational Scenario

Figure 8 depicts the AERO intradomain route optimization 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 ('R1'), two AERO
Servers ('S1', 'S2'), two AERO Clients ('C1', 'C2') and two ordinary
IPv6 hosts ('H1', 'H2'):

```
         +--------------+  +--------------+  +--------------+
         |   Server S1  |  |   Relay R1   |  |   Server S2  |
         +--------------+  +--------------+  +--------------+
            fe80::2             fe80::1            fe80::3
            L2(S1)              L2(R1)             L2(S2)
              |                   |                  |
    X-----+-----+-----------------+----------------+----+----X
          |           AERO Link                      |
        L2(A)                                       L2(B)
   fe80::2001:db8:0:0                        fe80::2001:db8:1:0
   +--------------+                           +--------------+
   |AERO Client C1|                           |AERO Client C2|
   +--------------+                           +--------------+
    2001:DB8:0::/48                            2001:DB8:1::/48
          |                                         |
        .-.                                       .-.
      ,-(  _)-.   2001:db8:0::1     2001:db8:1::1    ,-(  _)-.
    .-(_  IP   )-.   +---------+     +---------+    .-(_  IP   )-.
   (__    EUN      )--| Host H1 |     | Host H2 |--(__    EUN      )
     `-(_____)-'     +---------+     +---------+     `-(_____)-'
```
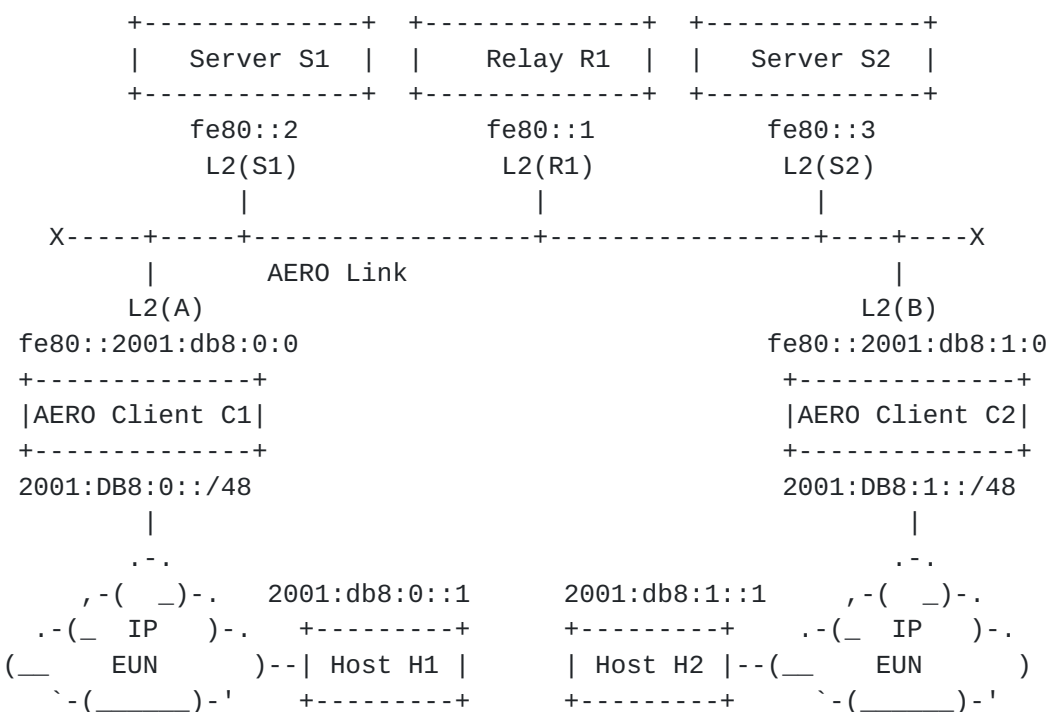
                 Figure 8: AERO Reference Operational Scenario

In Figure 8, Relay ('R1') assigns the address fe80::1 to its AERO
interface with link-layer address L2(R1), Server ('S1') assigns the
address fe80::2 with link-layer address L2(S1),and Server ('S2')
assigns 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 ('C1') receives the ACP 2001:db8:0::/48 in a DHCPv6 PD
exchange via AERO Server ('S1') then assigns the address
fe80::2001:db8:0:0 to its AERO interface with link-layer address
L2(C1). Client ('C1') 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 ('H1') connects to the EUN, and configures the
address 2001:db8:0::1.

AERO Client ('C2') receives the ACP 2001:db8:1::/48 in a DHCPv6 PD
exchange via AERO Server ('S2') then assigns the address
fe80::2001:db8:1:0 to its AERO interface with link-layer address
L2(C2). Client ('C2') 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 ('H1') connects to the EUN, and configures the
address 2001:db8:1::1.

[3.17.2](#). Concept of Operations

Again, with reference to Figure 8, when source host ('H1') sends a
packet to destination host ('H2'), the packet is first forwarded over
the source host's attached EUN to Client ('C1'). Client ('C1') then
forwards the packet via its AERO interface to Server ('S1') and also
sends a Predirect message toward Client ('C2') via Server ('S1').
Server ('S1') then re-encapsulates and forwards both the packet and
the Predirect message out the same AERO interface toward Client
('C2') via Relay ('R1').

When Relay ('R1') receives the packet and Predirect message, it
consults its forwarding table to discover Server ('S2') as the next
hop toward Client ('C2'). Relay ('R1') then forwards both the packet
and the Predirect message to Server ('S2'), which then forwards them
to Client ('C2').

After Client ('C2') receives the Predirect message, it process the
message and returns a Redirect message toward Client ('C1') via
Server ('S2'). During the process, Client ('C2') also creates or
updates a dynamic neighbor cache entry for Client ('C1').

When Server ('S2') receives the Redirect message, it re-encapsulates
the message and forwards it on to Relay ('R1'), which forwards the
message on to Server ('S1') which forwards the message on to Client
('C1'). After Client ('C1') receives the Redirect message, it
processes the message and creates or updates a dynamic neighbor cache
entry for Client ('C2').

Following the above Predirect/Redirect message exchange, forwarding
of packets from Client ('C1') to Client ('C2') without involving any
intermediate nodes is enabled. The mechanisms that support this
exchange are specified in the following sections.

3.17.3. 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 9:

```
     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 ...
    +-+-+-+-+-+-+-+-+-+-
```
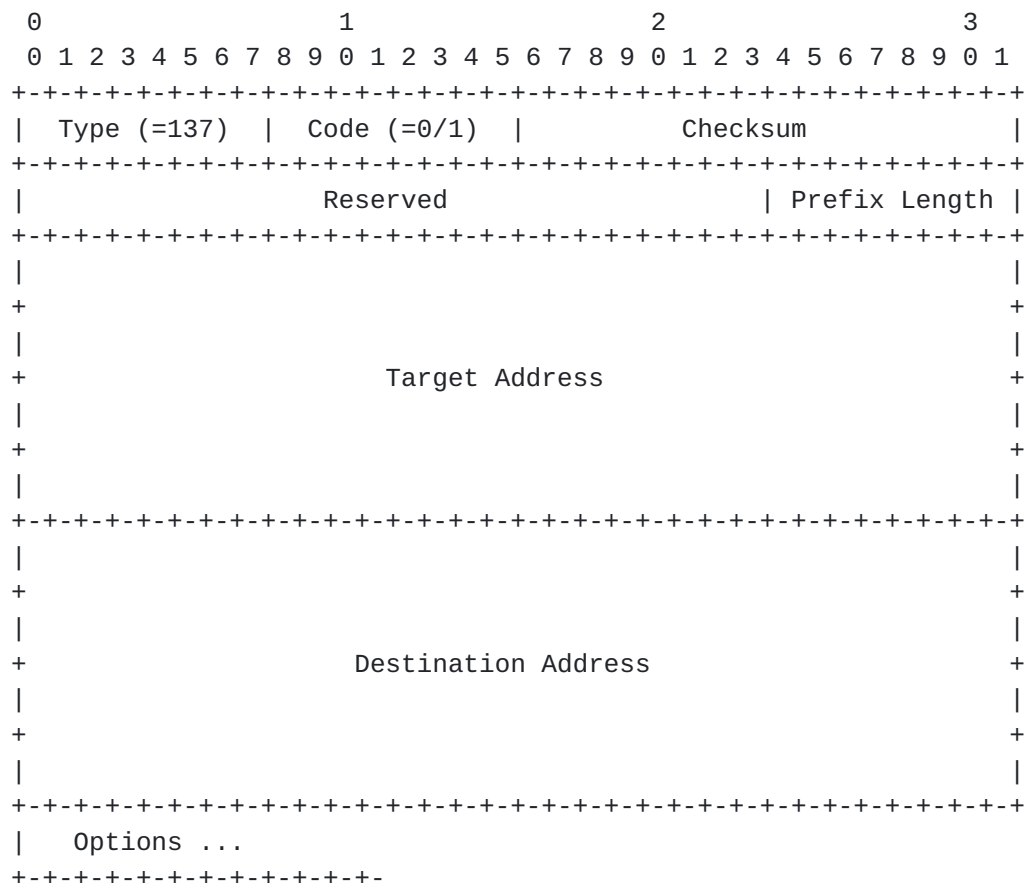
              Figure 9: AERO Redirect/Predirect Message Format

3.17.4. 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 ('C1') forwards a
packet toward Client ('C2'), it MAY also send a Predirect message

forward toward Client ('C2'), subject to rate limiting (see
[Section 8.2 of [RFC4861]](#)). Client ('C1') prepares the Predirect
message as follows:

o the link-layer source address is set to 'L2(C1)' (i.e., the link-
   layer address of Client ('C1')).

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

o the network-layer source address is set to fe80::2001:db8:0:0
   (i.e., the AERO address of Client ('C1')).

o the network-layer destination address is set to fe80::2001:db8:1:0
   (i.e., the AERO address of Client ('C2')).

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
   assigned to the Target Address.

o the Target Address is set to fe80::2001:db8:0:0 (i.e., the AERO
   address of Client ('C1')).

o 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).

o the message includes one or more TLLAOs with Link ID and DSCPs set
   to appropriate values for Client ('C1')'s underlying interfaces,
   and with UDP Port Number and IP Address set to 0'.

o the message SHOULD include a Timestamp option and a Nonce option.

o the message includes a Redirected Header Option (RHO) that
   contains the originating packet truncated if necessary 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 ('C1') may have advanced knowledge that the direct path
to Client ('C2') would be unusable or otherwise undesirable. If the
direct path later becomes unusable after the initial route
optimization, Client ('C1') simply allows packets to again flow
through Server ('S1').

3.17.5. Re-encapsulating and Relaying Predirects

   When Server ('S1') receives a Predirect message from Client ('C1'),
   it first verifies that the TLLAOs in the Predirect are a proper
   subset of the Link IDs in Client ('C1')'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 ('C1') 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 ('C1')'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 ('C2') by
   searching for the AERO address in the neighbor cache. Since Client
   ('C2') is not one of its neighbors, Server ('S1') re-encapsulates the
   Predirect and relays it via Relay ('R1') by changing the link-layer
   source address of the message to 'L2(S1)' and changing the link-layer
   destination address to 'L2(R1)'. Server ('S1') finally forwards the
   re-encapsulated message to Relay ('R1') without decrementing the
   network-layer TTL/Hop Limit field.

   When Relay ('R1') receives the Predirect message from Server ('S1')
   it determines that Server ('S2') is the next hop toward Client ('C2')
   by consulting its forwarding table. Relay ('R1') then re-
   encapsulates the Predirect while changing the link-layer source
   address to 'L2(R1)' and changing the link-layer destination address
   to 'L2(S2)'. Relay ('R1') then relays the Predirect via Server
   ('S2').

   When Server ('S2') receives the Predirect message from Relay ('R1')
   it determines that Client ('C2') 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(C2)'. Server ('S2') then
   forwards the message to Client ('C2').

3.17.6. Processing Predirects and Sending Redirects

   When Client ('C2') 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 ('C2') further accepts the
   message only if it is willing to serve as a redirection target.
   Next, Client ('C2') 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 ('C2') 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 ('C1') , stores the link-layer
addresses found in the TLLAOs as the link-layer addresses of Client
('C1') and stores the Prefix Length as the length to be applied to
the network-layer address for forwarding purposes. Client ('C2')
then sets AcceptTime for the neighbor cache entry to ACCEPT_TIME.

After processing the message, Client ('C2') prepares a Redirect
message response as follows:

o the link-layer source address is set to 'L2(C2)' (i.e., the link-
   layer address of Client ('C2')).

o the link-layer destination address is set to 'L2(S2)' (i.e., the
   link-layer address of Server ('S2')).

o the network-layer source address is set to fe80::2001:db8:1:0
   (i.e., the AERO address of Client ('C2')).

o the network-layer destination address is set to fe80::2001:db8:0:0
   (i.e., the AERO address of Client ('C1')).

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 Address.

o the Target Address is set to fe80::2001:db8:1:0 (i.e., the AERO
   address of Client ('C2')).

o 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).

o the message includes one or more TLLAOs with Link ID and DSCPs set
   to appropriate values for Client ('C2')'s underlying interfaces,
   and with UDP Port Number and IP Address set to '0'.

   o 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).

   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 ('C2') prepares the Redirect message, it sends the
   message to Server ('S2').

3.17.7. Re-encapsulating and Relaying Redirects

   When Server ('S2') receives a Redirect message from Client ('C2'), it
   first verifies that the TLLAOs in the Redirect are a proper subset of
   the Link IDs in Client ('C2')'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 ('C2') 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 ('C2')'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 ('C2') by
   searching for the AERO address in the neighbor cache. Since Client
   ('C2') is not a neighbor, Server ('S2') re-encapsulates the Predirect
   and relays it via Relay ('R1') by changing the link-layer source
   address of the message to 'L2(S2)' and changing the link-layer
   destination address to 'L2(R1)'. Server ('S2') finally forwards the
   re-encapsulated message to Relay ('R1') without decrementing the
   network-layer TTL/Hop Limit field.

   When Relay ('R1') receives the Predirect message from Server ('S2')
   it determines that Server ('S1') is the next hop toward Client ('C1')
   by consulting its forwarding table. Relay ('R1') then re-
   encapsulates the Predirect while changing the link-layer source
   address to 'L2(R1)' and changing the link-layer destination address
   to 'L2(S1)'. Relay ('R1') then relays the Predirect via Server
   ('S1').

   When Server ('S1') receives the Predirect message from Relay ('R1')
   it determines that Client ('C1') 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(C1)'. Server ('S1') then
forwards the message to Client ('C1').

3.17.8. Processing Redirects

When Client ('C1') 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 ('C1') 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 ('C1') then processes
the message as follows.

In the reference operational scenario, when Client ('C1') 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 ('C2'), stores the link-layer
addresses found in the TLLAOs as the link-layer addresses of Client
('C2') and stores the Prefix Length as the length to be applied to
the network-layer address for forwarding purposes. Client ('C1')
then sets ForwardTime for the neighbor cache entry to FORWARD_TIME.

Now, Client ('C1') has a neighbor cache entry with a valid
ForwardTime value, while Client ('C2') has a neighbor cache entry
with a valid AcceptTime value. Thereafter, Client ('C1') may forward
ordinary network-layer data packets directly to Client ('C2') without
involving any intermediate nodes, and Client ('C2') can verify that
the packets came from an acceptable source. (In order for Client
('C2') to forward packets to Client ('C1'), a corresponding
Predirect/Redirect message exchange is required in the reverse
direction; hence, the mechanism is asymmetric.)

3.17.9. 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 3.17.6), except that it writes its own link-layer
address in the TLLAO option. The Server must then maintain a dynamic
neighbor cache entry for the redirected source Client.

3.18. Neighbor Unreachability Detection (NUD)

   AERO nodes perform Neighbor Unreachability Detection (NUD) by sending
   unicast NS messages to elicit solicited NA messages from neighbors
   the same as described in [RFC4861]. NUD is performed either
   reactively in response to persistent L2 errors (see Section 3.14) or
   proactively to refresh existing neighbor cache entries.

   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
   proactively 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 proactively
   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.

3.19. Mobility Management

3.19.1. 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 address and with an ALREQ that includes the correct Link ID
and DSCP values. If authentication succeeds, the Server then update
its neighbor cache and sends a DHCPv6 Reply. Note that if the Client
does not issue a DHCPv6 Rebind before the prefix delegation lifetime
expires (e.g., if the Client has been out of touch with the Server
for a considerable amount of time), the Server's Reply will report
NoBinding and the Client must re-initiate the DHCPv6 PD procedure.

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 8, when Client ('C2') needs to change its
link-layer address it sends unicast unsolicited NA messages to Client
('C1') via Server ('S2') as follows:

o the link-layer source address is set to 'L2(C2)' (i.e., the link-
   layer address of Client ('C2')).

o the link-layer destination address is set to 'L2(S2)' (i.e., the
   link-layer address of Server ('S2')).

o the network-layer source address is set to fe80::2001:db8:1:0
   (i.e., the AERO address of Client ('C2')).

o the network-layer destination address is set to fe80::2001:db8:0:0
   (i.e., the AERO address of Client ('C1')).

o   the Type is set to 136.

o   the Code is set to 0.

o   the Solicited flag is set to 0.

o   the Override flag is set to 1.

o   the Target Address is set to fe80::2001:db8:1:0 (i.e., the AERO
    address of Client ('C2')).

o   the message includes one or more TLLAOs with Link ID and DSCPs set
    to appropriate values for Client ('C2')'s underlying interfaces,
    and with UDP Port Number and IP Address set to '0'.

o   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 3.17.7. 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 ('R1'), then forwards the NA message
via the relaying chain the same as for a Redirect.

When Client ('C1') receives the NA message, it accepts the message
only if it already has a neighbor cache entry for Client ('C2') then
updates the link-layer addresses for Client ('C2') based on the
addresses in the TLLAOs. Next, Client ('C1') SHOULD initiate the NUD
procedures specified in Section 3.18 to provide Client ('C2') with an
indication that the link-layer source address has been updated, and
to refresh ('C2')'s AcceptTime and ('C1')'s ForwardTime timers.

If Client ('C2') receives an NS message from Client ('C1') indicating
that an unsolicited NA has updated its neighbor cache, Client ('C2')
need not send additional unsolicited NAs. If Client ('C2')'s
unsolicited NA messages are somehow lost, however, Client ('C1') will
soon learn of the mobility event via NUD.

3.19.2. 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 an ALREQ that includes
the new Link ID and DSCP 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 3.19.1.

3.19.3. 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 an ALDEL that includes the deprecated
Link ID. 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 3.19.1.

3.19.4. Moving to a New Server

When a Client associates with a new Server, it performs the Client
procedures specified in Section 3.15.2.

When a Client disassociates with an existing Server, it sends a
DHCPv6 Release message via a new Server to the unicast link-local
network layer address of the old Server. The new Server then writes
its own link-layer address in the DHCPv6 Release message IP source
address and forwards the message to the old Server.

When the old Server receives the DHCPv6 Release, it first
authenticates the message. The Server then resets the Client's
neighbor cache entry lifetime to 5 seconds, rewrites the link-layer
address in the neighbor cache entry to the address of the new Server,
then returns a DHCPv6 Reply message to the Client via the old Server.
When the lifetime expires, the old Server withdraws the IP route from
the AERO routing system and deletes the neighbor cache entry for the
Client. 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. (Note
that since Release/Reply messages may be lost in the network the
Client MUST retry until it gets Reply indicating that the Release was
successful.)

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

3.20. Proxy AERO

   Proxy Mobile IPv6 (PMIPv6) [RFC5213][RFC5844][RFC5949] presents a
   localized mobility management scheme for use within an access network
   domain. It is typically used in WiFi and cellular wireless access
   networks, and allows Mobile Nodes (MNs) to receive and retain an IP
   address that remains stable within the access network domain without
   needing to implement any special mobility protocols. In the PMIPv6
   architecture, access network devices known as Mobility Access
   Gateways (MAGs) provide MNs with an access link abstraction and
   receive prefixes for the MNs from a Local Mobility Anchor (LMA).

   In a proxy AERO domain, a proxy AERO Client (acting as a MAG) can
   similarly provide proxy services for MNs that do not participate in
   AERO messaging. The proxy Client presents an access link abstraction
   to MNs, and performs DHCPv6 PD exchanges over the AERO interface with
   an AERO Server (acting as an LMA) to receive ACPs for address
   provisioning of new MNs that come onto an access link. This scheme
   assumes that proxy Clients act as fixed (non-mobile) infrastructure
   elements under the same administrative trust basis as for Relays and
   Servers.

   When an MN comes onto an access link within a proxy AERO domain for
   the first time, the proxy Client authenticates the MN and obtains a
   unique identifier that it can use as a DHCPv6 DUID then issues a
   DHCPv6 PD Solicit/Request to its Server. When the Server delegates
   an ACP, the proxy Client creates an AERO address for the MN and
   assigns the ACP to the MN's access link. The proxy Client then
   configures itself as a default router for the MN and provides address
   autoconfiguration services (e.g., SLAAC, DHCPv6, DHCPv4, etc.) for
   provisioning MN addresses from the ACP over the access link. Since
   the proxy Client may serve many such MNs simultaneously, it may
   receive multiple ACP prefix delegations and configure multiple AERO
   addresses, i.e., one for each MN.

   When two MNs are associated with the same proxy Client, the Client
   can forward traffic between the MNs without involving a Server since
   it configures the AERO addresses of both MNs and therefore also has
   the necessary routing information. When two MNs are associated with
   different proxy Clients, the source MN's Client can initiate standard
   AERO route optimization to discover a direct path to the target MN's
   Client through the exchange of Predirect/Redirect messages.

   When an MN in a proxy AERO domain leaves an access link provided by
   an old proxy Client, the MN issues an access link-specific "leave"
   message that informs the old Client of the link-layer address of a
   new Client on the planned new access link. This is known as a
   "predictive handover". When an MN comes onto an access link provided

by a new proxy Client, the MN issues an access link-specific "join"
message that informs the new Client of the link-layer address of the
old Client on the actual old access link. This is known as a
"reactive handover".

Upon receiving a predictive handover indication, the old proxy Client
sends a DHCPv6 PD Solicit/Request message directly to the new Client
and queues any arriving data packets addressed to the departed MN.
The Solicit/Request message includes the MN's ID as the DUID, the ACP
in an IA_PD option, the old Client's address as the link-layer source
address and the new Client's address as the link-layer destination
address. When the new Client receives the Solicit/Request message,
it changes the link-layer source address to its own address, changes
the link-layer destination address to the address of its Server, and
forwards the message to the Server. At the same time, the new Client
creates access link state for the ACP in anticipation of the MN's
arrival (while queuing any data packets until the MN arrives),
creates a neighbor cache entry for the old Client with AcceptTime set
to ACCEPT_TIME, then sends a Redirect message back to the old Client.
When the old Client receives the Redirect message, it creates a
neighbor cache entry for the new Client with ForwardTime set to
FORWARD_TIME, then forwards any queued data packets to the new
Client. At the same time, the old Client sends a DHCPv6 PD Release
message to its Server. Finally, the old Client sends unsolicited NA
messages to any of the ACP's correspondents with a TLLAO containing
the link-layer address of the new Client. This follows the procedure
specified in Section 3.19.1, except that it is the old Client and not
the Server that supplies the link-layer address.

Upon receiving a reactive handover indication, the new proxy Client
creates access link state for the MN's ACP, sends a DHCPv6 PD
Solicit/Request message to its Server, and sends a DHCPv6 PD Release
message directly to the old Client. The Release message includes the
MN's ID as the DUID, the ACP in an IA_PD option, the new Client's
address as the link-layer source address and the old Client's address
as the link-layer destination address. When the old Client receives
the Release message, it changes the link-layer source address to its
own address, changes the link-layer destination address to the
address of its Server, and forwards the message to the Server. At
the same time, the old Client sends a Predirect message back to the
new Client and queues any arriving data packets addressed to the
departed MN. When the new Client receives the Predirect, it creates
a neighbor cache entry for the old Client with AcceptTime set to
ACCEPT_TIME, then sends a Redirect message back to the old Client.
When the old Client receives the Redirect message, it creates a
neighbor cache entry for the new Client with ForwardTime set to
FORWARD_TIME, then forwards any queued data packets to the new

Client. Finally, the old Client sends unsolicited NA messages to
correspondents the same as for the predictive case.

When a Server processes a DHCPv6 Solicit/Request message, it creates
a neighbor cache entry for this ACP if none currently exists. If a
neighbor cache entry already exists, however, the Server changes the
link-layer address to the address of the new proxy Client (this
satisfies the case of both the old Client and new Client using the
same Server).

When a Server processes a DHCPv6 Release message, it resets the
neighbor cache entry lifetime for this ACP to 5 seconds if the cached
link-layer address matches the old proxy Client's address.
Otherwise, the Server ignores the Release message (this satisfies the
case of both the old Client and new Client using the same Server).

When a correspondent Client receives an unsolicited NA message, it
changes the link-layer address for the ACP's neighbor cache entry to
the address of the new proxy Client. The correspondent Client then
issues a Predirect/Redirect exchange to establish a new neighbor
cache entry in the new Client.

From an architectural perspective, in addition to the use of DHCPv6
PD and IPv6 ND signaling the AERO approach differs from PMIPv6 in its
use of the NBMA virtual link model instead of point-to-point tunnels.
This provides a more agile interface for Client/Server and Client/
Client coordinations, and also facilitates simple route optimization.
The AERO routing system is also arranged in such a fashion that
Clients get the same service from any Server they happen to associate
with. This provides a natural fault tolerance and load balancing
capability such as desired for distributed mobility management.

3.21. Extending AERO Links Through Security Gateways

When an enterprise mobile device moves from a campus LAN connection
to a public Internet link, it must re-enter the enterprise via a
security gateway that has both a physical interface connection to the
Internet and a physical interface connection to the enterprise
internetwork. This most often entails the establishment of a Virtual
Private Network (VPN) link over the public Internet from the mobile
device to the security gateway. During this process, the mobile
device supplies the security gateway with its public Internet address
as the link-layer address for the VPN. The mobile device then acts
as an AERO Client to negotiate with the security gateway to obtain
its ACP.

In order to satisfy this need, the security gateway also operates as
an AERO Server with support for AERO Client proxying. In particular,

when a mobile device (i.e., the Client) connects via the security
gateway (i.e., the Server), the Server provides the Client with an
ACP in a DHCPv6 PD exchange the same as if it were attached to an
enterprise campus access link. The Server then replaces the Client's
link-layer source address with the Server's enterprise-facing link-
layer address in all AERO messages the Client sends toward neighbors
on the AERO link. The AERO messages are then delivered to other
devices on the AERO link as if they were originated by the security
gateway instead of by the AERO Client. In the reverse direction, the
AERO messages sourced by devices within the enterprise network can be
forwarded to the security gateway, which then replaces the link-layer
destination address with the Client's link-layer address and replaces
the link-layer source address with its own (Internet-facing) link-
layer address.

After receiving the ACP, the Client can send IP packets that use an
address taken from the ACP as the network layer source address, the
Client's link-layer address as the link-layer source address, and the
Server's Internet-facing link-layer address as the link-layer
destination address. The Server will then rewrite the link-layer
source address with the Server's own enterprise-facing link-layer
address and rewrite the link-layer destination address with the
target AERO node's link-layer address, and the packets will enter the
enterprise network as though they were sourced from a device located
within the enterprise. In the reverse direction, when a packet
sourced by a node within the enterprise network uses a destination
address from the Client's ACP, the packet will be delivered to the
security gateway which then rewrites the link-layer destination
address to the Client's link-layer address and rewrites the link-
layer source address to the Server's Internet-facing link-layer
address. The Server then delivers the packet across the VPN to the
AERO Client. In this way, the AERO virtual link is essentially
extended *through* the security gateway to the point at which the VPN
link and AERO link are effectively grafted together by the link-layer
address rewriting performed by the security gateway. All AERO
messaging services (including route optimization and mobility
signaling) are therefore extended to the Client.

In order to support this virtual link grafting, the security gateway
(acting as an AERO Server) must keep static neighbor cache entries
for all of its associated Clients located on the public Internet.
The neighbor cache entry is keyed by the AERO Client's AERO address
the same as if the Client were located within the enterprise
internetwork. The neighbor cache is then managed in all ways as
though the Client were an ordinary AERO Client. This includes the
AERO IPv6 ND messaging signaling for Route Optimization and Neighbor
Unreachability Detection.

Note that the main difference between a security gateway acting as an
AERO Server and an enterprise-internal AERO Server is that the
security gateway has at least one enterprise-internal physical
interface and at least one public Internet physical interface.
Conversely, the enterprise-internal AERO Server has only enterprise-
internal physical interfaces. For this reason security gateway
proxying is needed to ensure that the public Internet link-layer
addressing space is kept separate from the enterprise-internal link-
layer addressing space. This is afforded through a natural extension
of the security association caching already performed for each VPN
client by the security gateway.

3.22. Extending IPv6 AERO Links to the Internet

When an IPv6 host ('H1') with an address from an ACP owned by AERO
Client ('C1') sends packets to a correspondent IPv6 host ('H2'), the
packets eventually arrive at the IPv6 router that owns ('H2')s
prefix. This IPv6 router may or may not be an AERO Client ('C2')
either within the same home network as ('C1') or in a different home
network.

If Client ('C1') is currently located outside the boundaries of its
home network, it will connect back into the home network via a
security gateway acting as an AERO Server. The packets sent by
('H1') via ('C1') will then be forwarded through the security gateway
then through the home network and finally to ('C2') where they will
be delivered to ('H2'). This could lead to sub-optimal performance
when ('C2') could instead be reached via a more direct route without
involving the security gateway.

Consider the case when host ('H1') has the IPv6 address
2001:db8:1::1, and Client ('C1') has the ACP 2001:db8:1::/64 with
underlying IPv6 Internet address of 2001:db8:1000::1. Also, host
('H2') has the IPv6 address 2001:db8:2::1, and Client ('C2') has the
ACP 2001:db8:2::/64 with underlying IPv6 address of 2001:db8:2000::1.
Client ('C1') can determine whether 'C2' is indeed also an AERO
Client willing to serve as a route optimization correspondent by
resolving the AAAA records for the DNS FQDN that matches ('H2')s
prefix, i.e.:

'0.0.0.0.2.0.0.0.8.b.d.0.1.0.0.2.aero.linkupnetworks.net'

If ('C2') is indeed a candidate correspondent, the FQDN lookup will
return a PTR resource record that contains the domain name for the
AERO link that manages ('C2')s ASP. Client ('C1') can then attempt
route optimization using an approach similar to the Return
Routability procedure specified for Mobile IPv6 (MIPv6) [RFC6275].
In order to support this process, both Clients MUST intercept and

decapsulate packets that have a subnet router anycast address
corresponding to any of the /64 prefixes covered by their respective
ACPs.

To initiate the process, Client ('C1') creates a specially-crafted
encapsulated AERO Predirect message that will be routed through its
home network then through ('C2')s home network and finally to ('C2')
itself. Client ('C1') prepares the initial message in the exchange
as follows:

o The encapsulating IPv6 header source address is set to
   2001:db8:1:: (i.e., the IPv6 subnet router anycast address for
   ('C1')s ACP)

o The encapsulating IPv6 header destination address is set to
   2001:db8:2:: (i.e., the IPv6 subnet router anycast address for
   ('C2')s ACP)

o The encapsulating IPv6 header is followed by a UDP header with
   source and destination port set to 8060

o The encapsulated IPv6 header source address is set to
   fe80::2001:db8:1:0 (i.e., the AERO address for ('C1'))

o The encapsulated IPv6 header destination address is set to
   fe80::2001:db8:2:0 (i.e., the AERO address for ('C2'))

o The encapsulated AERO Predirect message includes all of the
   securing information that would occur in a MIPv6 "Home Test Init"
   message (format TBD)

Client ('C1') then further encapsulates the message in the
encapsulating headers necessary to convey the packet to the security
gateway (e.g., through IPsec encapsulation) so that the message now
appears "double-encapsulated". ('C1') then sends the message to the
security gateway, which re-encapsulates and forwards it over the home
network from where it will eventually reach ('C2').

At the same time, ('C1') creates and sends a second encapsulated AERO
Predirect message that will be routed through the IPv6 Internet
without involving the security gateway. Client ('C1') prepares the
message as follows:

o The encapsulating IPv6 header source address is set to
   2001:db8:1000:1 (i.e., the Internet IPv6 address of ('C1'))

   o The encapsulating IPv6 header destination address is set to
      2001:db8:2:: (i.e., the IPv6 subnet router anycast address for
      ('C2')s ACP)

   o The encapsulating IPv6 header is followed by a UDP header with
      source and destination port set to 8060

   o The encapsulated IPv6 header source address is set to
      fe80::2001:db8:1:0 (i.e., the AERO address for ('C1'))

   o The encapsulated IPv6 header destination address is set to
      fe80::2001:db8:2:0 (i.e., the AERO address for ('C2'))

   o The encapsulated AERO Predirect message includes all of the
      securing information that would occur in a MIPv6 "Care-of Test
      Init" message (format TBD)

   ('C2') will receive both Predirect messages through its home network
   then return a corresponding Redirect for each of the Predirect
   messages with the source and destination addresses in the inner and
   outer headers reversed. The first message includes all of the
   securing information that would occur in a MIPv6 "Home Test" message,
   while the second message includes all of the securing information
   that would occur in a MIPv6 "Care-of Test" message (formats TBD).

   When ('C1') receives the Redirect messages, it performs the necessary
   security procedures per the MIPv6 specification. It then prepares an
   encapsulated NS message that includes the same source and destination
   addresses as for the "Care-of Test Init" Predirect message, and
   includes all of the securing information that would occur in a MIPv6
   "Binding Update" message (format TBD) and sends the message to
   ('C2').

   When ('C2') receives the NS message, if the securing information is
   correct it creates or updates a neighbor cache entry for ('C1') with
   fe80::2001:db8:1:0 as the network-layer address, 2001:db8:1000::1 as
   the link-layer address and with AcceptTime set to ACCEPT_TIME.
   ('C2') then sends an encapsulated NA message back to ('C1') that
   includes the same source and destination addresses as for the "Care-
   of Test" Redirect message, and includes all of the securing
   information that would occur in a MIPv6 "Binding Acknowledgement"
   message (format TBD) and sends the message to ('C1').

   When ('C1') receives the NA message, it creates or updates a neighbor
   cache entry for ('C2') with fe80::2001:db8:2:0 as the network-layer
   address and 2001:db8:2:: as the link-layer address and with
   ForwardTime set to FORWARD_TIME, thus completing the route
   optimization in the forward direction.

   ('C1') subsequently forwards encapsulated packets with outer source
   address 2001:db8:1000::1, with outer destination address
   2001:db8:2::, with inner source address taken from the 2001:db8:1::,
   and with inner destination address taken from 2001:db8:2:: due to the
   fact that it has a securely-established neighbor cache entry with
   non-zero ForwardTime. ('C2') subsequently accepts any such
   encapsulated packets due to the fact that it has a securely-
   established neighbor cache entry with non-zero AcceptTime.

   In order to keep neighbor cache entries alive, ('C1') periodically
   sends additional NS messages to ('C2') and receives any NA responses.
   If ('C1') moves to a different point of attachment after the initial
   route optimization, it sends a new secured NS message to ('C2') as
   above to update ('C2')s neighbor cache.

   If ('C2') has packets to send to ('C1'), it performs a corresponding
   route optimization in the opposite direction following the same
   procedures described above. In the process, the already-established
   unidirectional neighbor cache entries within ('C1') and ('C2') are
   updated to include the now-bidirectional information. In particular,
   the AcceptTime and ForwardTime variables for both neighbor cache
   entries are updated to non-zero values, and the link-layer address
   for ('C1')s neighbor cache entry for ('C2') is reset to
   2001:db8:2000::1.

   Note that two AERO Clients can use full security protocol messaging
   instead of Return Routability, e.g., if strong authentication and/or
   confidentiality are desired. In that case, security protocol key
   exchanges such as specified for MOBIKE [RFC4555] would be used to
   establish security associations and neighbor cache entries between
   the AERO clients. Thereafter, AERO NS/NA messaging can be used to
   maintain neighbor cache entries, test reachability, and to announce
   mobility events. If reachability testing fails, e.g., if both
   Clients move at roughly the same time, the Clients can tear down the
   security association and neighbor cache entries and again allow
   packets to flow through their home network.

3.23. 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.

3.24. 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".)

3.25. 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.

3.26. 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.

3.27. Manually-Configured AERO Tunnels

   In addition to the dynamic neighbor discovery procedures for AERO
   link neighbors described above, AERO encapsulation can be applied to
   manually-configured tunnels. In that case, the tunnel endpoints use
   an administratively-assigned link-local address and exchange NS/NA
   messages the same as for dynamically-established tunnels.

3.28. Intradomain Routing

   After a tunnel neighbor relationship has been established, neighbors
   can use a traditional dynamic routing protocol over the tunnel to
   exchange routing information without having to inject the routes into
   the AERO routing system.

4. Implementation Status

   User-level and kernel-level AERO implementations have been developed
   and are undergoing internal testing within Boeing.

5. Next Steps

   A new Generic UDP Encapsulation (GUE) format has been specified in
   [I-D.herbert-gue-fragmentation] [I-D.ietf-nvo3-gue]. The GUE
   encapsulation format will eventually supplant the native AERO UDP
   encapsulation format.

   Future versions of the spec will explore the subject of DSCP marking
   in more detail.

6. IANA Considerations

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

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

   No further IANA actions are required.

7. 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. Unless there is some other means
   of authenticating the Client's identity (e.g., link-layer security),
   AERO nodes SHOULD also use DHCPv6 securing services (e.g., DHCPv6
   authentication, Secure DHCPv6 [I-D.ietf-dhc-sedhcpv6], etc.) for
   Client authentication and network admission control. In particular,
   Clients SHOULD include authenticating information on each
   Solicit/Request/Rebind/Release message they send, but omit
   authenticating information on Renew messages. Renew messages are
   exempt due to the fact that the Renew must already be checked for

having a correct link-layer address and does not update any link-
layer addresses. Therefore, asking the Server to also authenticate
the Renew message would be unnecessary and could result in excessive
processing overhead.

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.

An AERO Client's link-layer address could be rewritten by a link-
layer switching element on the path from the Client to the Server and
not detected by the DHCPv6 security mechanism. However, such a
condition would only be a matter of concern on unmanaged/unsecured
links where the link-layer switching elements themselves present a
man-in-the-middle attack threat. For this reason, IP security MUST
be used when AERO is employed over unmanaged/unsecured links.

8. 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, Mark Andrews, Fred Baker,
Stewart Bryant, Brian Carpenter, Wojciech Dec, Ralph Droms, Adrian
Farrel, Sri Gundavelli, Brian Haberman, Joel Halpern, Tom Herbert,

Sascha Hlusiak, Lee Howard, Andre Kostur, Ted Lemon, Andy Malis,
Satoru Matsushima, Tomek Mrugalski, Alexandru Petrescu, Behcet
Saikaya, Joe Touch, Bernie Volz, Ryuji Wakikawa and Lloyd Wood.
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 Dave Bernhardt, Cam Brodie, Balaguruna
Chidambaram, Bruce Cornish, Claudiu Danilov, Wen Fang, Anthony
Gregory, Jeff Holland, Ed King, Gen MacLean, Rob Muszkiewicz, Sean
O'Sullivan, Kent Shuey, Brian Skeen, Mike Slane, Brendan Williams,
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].

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

o The Internet Routing Overlay Network (IRON)
   [RFC6179][I-D.templin-ironbis]

o Virtual Enterprise Traversal (VET)
   [RFC5558][I-D.templin-intarea-vet]

o The Subnetwork Encapsulation and Adaptation Layer (SEAL)
   [RFC5320][I-D.templin-intarea-seal]

o AERO, First Edition [RFC6706]

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

9. References

9.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
           DOI 10.17487/RFC0768, August 1980,
           <http://www.rfc-editor.org/info/rfc768>.

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

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

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

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

   [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6
              (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460,
              December 1998, <http://www.rfc-editor.org/info/rfc2460>.

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

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

   [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins,
              C., and M. Carney, "Dynamic Host Configuration Protocol
              for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July
              2003, <http://www.rfc-editor.org/info/rfc3315>.

   [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic
              Host Configuration Protocol (DHCP) version 6", RFC 3633,
              DOI 10.17487/RFC3633, December 2003,
              <http://www.rfc-editor.org/info/rfc3633>.

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

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

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

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

   [RFC6434] Jankiewicz, E., Loughney, J., and T. Narten, "IPv6 Node
             Requirements", RFC 6434, DOI 10.17487/RFC6434, December
             2011, <http://www.rfc-editor.org/info/rfc6434>.

9.2. Informative References

   [I-D.herbert-gue-fragmentation]
             Herbert, T. and F. Templin, "Fragmentation option for
             Generic UDP Encapsulation", draft-herbert-gue-
             fragmentation-00 (work in progress), March 2015.

   [I-D.ietf-dhc-sedhcpv6]
             Jiang, S., Shen, S., Zhang, D., and T. Jinmei, "Secure
             DHCPv6", draft-ietf-dhc-sedhcpv6-08 (work in progress),
             June 2015.

   [I-D.ietf-nvo3-gue]
             Herbert, T., Yong, L., and O. Zia, "Generic UDP
             Encapsulation", draft-ietf-nvo3-gue-01 (work in progress),
             June 2015.

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

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

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

[I-D.vandevelde-idr-remote-next-hop]
          Velde, G., Patel, K., Rao, D., Raszuk, R., and R. Bush,
          "BGP Remote-Next-Hop", draft-vandevelde-idr-remote-next-
          hop-09 (work in progress), March 2015.

[RFC0879] Postel, J., "The TCP Maximum Segment Size and Related
           Topics", RFC 879, DOI 10.17487/RFC0879, November 1983,
           <http://www.rfc-editor.org/info/rfc879>.

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

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

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

[RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation,
           selection, and registration of an Autonomous System (AS)",
           BCP 6, RFC 1930, DOI 10.17487/RFC1930, March 1996,
           <http://www.rfc-editor.org/info/rfc1930>.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery
           for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August
           1996, <http://www.rfc-editor.org/info/rfc1981>.

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

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

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms",
           RFC 2675, DOI 10.17487/RFC2675, August 1999,
           <http://www.rfc-editor.org/info/rfc2675>.

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

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

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

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

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

   [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi,
              "DNS Extensions to Support IP Version 6", RFC 3596,
              DOI 10.17487/RFC3596, October 2003,
              <http://www.rfc-editor.org/info/rfc3596>.

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

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

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

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

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

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

   [RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol
              (MOBIKE)", RFC 4555, DOI 10.17487/RFC4555, June 2006,
              <http://www.rfc-editor.org/info/rfc4555>.

   [RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name
              System", RFC 4592, DOI 10.17487/RFC4592, July 2006,
              <http://www.rfc-editor.org/info/rfc4592>.

   [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU
              Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007,
              <http://www.rfc-editor.org/info/rfc4821>.

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

   [RFC4994] Zeng, S., Volz, B., Kinnear, K., and J. Brzozowski,
              "DHCPv6 Relay Agent Echo Request Option", RFC 4994,
              DOI 10.17487/RFC4994, September 2007,
              <http://www.rfc-editor.org/info/rfc4994>.

   [RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V.,
              Chowdhury, K., and B. Patil, "Proxy Mobile IPv6",
              RFC 5213, DOI 10.17487/RFC5213, August 2008,
              <http://www.rfc-editor.org/info/rfc5213>.

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

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <http://www.rfc-editor.org/info/rfc5246>.

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

   [RFC5494] Arkko, J. and C. Pignataro, "IANA Allocation Guidelines
              for the Address Resolution Protocol (ARP)", RFC 5494,
              DOI 10.17487/RFC5494, April 2009,
              <http://www.rfc-editor.org/info/rfc5494>.

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

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

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

   [RFC5720] Templin, F., "Routing and Addressing in Networks with
              Global Enterprise Recursion (RANGER)", RFC 5720,
              DOI 10.17487/RFC5720, February 2010,
              <http://www.rfc-editor.org/info/rfc5720>.

   [RFC5844] Wakikawa, R. and S. Gundavelli, "IPv4 Support for Proxy
              Mobile IPv6", RFC 5844, DOI 10.17487/RFC5844, May 2010,
              <http://www.rfc-editor.org/info/rfc5844>.

   [RFC5949] Yokota, H., Chowdhury, K., Koodli, R., Patil, B., and F.
              Xia, "Fast Handovers for Proxy Mobile IPv6", RFC 5949,
              DOI 10.17487/RFC5949, September 2010,
              <http://www.rfc-editor.org/info/rfc5949>.

   [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen,
              "Internet Key Exchange Protocol Version 2 (IKEv2)",
              RFC 5996, DOI 10.17487/RFC5996, September 2010,
              <http://www.rfc-editor.org/info/rfc5996>.

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

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

   [RFC6204] Singh, H., Beebee, W., Donley, C., Stark, B., and O.
              Troan, Ed., "Basic Requirements for IPv6 Customer Edge
              Routers", RFC 6204, DOI 10.17487/RFC6204, April 2011,
              <http://www.rfc-editor.org/info/rfc6204>.

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

   [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
              and A. Bierman, Ed., "Network Configuration Protocol
              (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
              <http://www.rfc-editor.org/info/rfc6241>.

   [RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility
              Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July
              2011, <http://www.rfc-editor.org/info/rfc6275>.

   [RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based
              DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355,
              DOI 10.17487/RFC6355, August 2011,
              <http://www.rfc-editor.org/info/rfc6355>.

   [RFC6422] Lemon, T. and Q. Wu, "Relay-Supplied DHCP Options",
              RFC 6422, DOI 10.17487/RFC6422, December 2011,
              <http://www.rfc-editor.org/info/rfc6422>.

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

   [RFC6691] Borman, D., "TCP Options and Maximum Segment Size (MSS)",
              RFC 6691, DOI 10.17487/RFC6691, July 2012,
              <http://www.rfc-editor.org/info/rfc6691>.

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

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

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

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

   [RFC6939] Halwasia, G., Bhandari, S., and W. Dec, "Client Link-Layer
              Address Option in DHCPv6", RFC 6939, DOI 10.17487/RFC6939,
              May 2013, <http://www.rfc-editor.org/info/rfc6939>.

   [RFC6980] Gont, F., "Security Implications of IPv6 Fragmentation
              with IPv6 Neighbor Discovery", RFC 6980,
              DOI 10.17487/RFC6980, August 2013,
              <http://www.rfc-editor.org/info/rfc6980>.

   [RFC7078] Matsumoto, A., Fujisaki, T., and T. Chown, "Distributing
              Address Selection Policy Using DHCPv6", RFC 7078,
              DOI 10.17487/RFC7078, January 2014,
              <http://www.rfc-editor.org/info/rfc7078>.

   [TUNTAP]   Wikipedia, W., "http://en.wikipedia.org/wiki/TUN/TAP",
              October 2014.

Author's Address

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

   Email: fltemplin@acm.org