Stateless user-plane architecture for virtualized EPC (vEPC)
draft-matsushima-stateless-uplane-vepc-04

Abstract

We envision a new mobile architecture for the future Evolved Packet Core (EPC). The new architecture is designed to support the virtualization scheme called NFV (Network Function Virtualization). In our architecture, the user plane of EPC is decoupled from the control-plane and uses routing information to forward packets of mobile nodes. Although the EPC control plane will run on hypervisor, our proposal does not modify the signaling of the EPC control plane. The benefits of our architecture are 1) scalability, 2) flexibility and 3) Manageability. How to run the EPC control plane on NFV is out of our focus in this document.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 11, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of
1. Introduction

3GPP introduces Evolved Packet Core (EPC) that is fully IP based mobile system for LTE and -advanced in their Release-8 specification and beyond. Operators are now deploying EPC for LTE services and encounter rapid LTE traffic growth. There are various activities to offload mobile traffic in 3GPP and IETF such as LIPA, SIPTO and DMM. The concept is similar that traffic of OTT (Over The Top) application is offloaded at entity that is closer to the mobile node (ex. eNodeB or closer anchor).

Likewise, overload of signaling (control plane) is also increasing day by day. Network operators expect recent innovation and trends of NFV (Network Function Virtualization) to solve this overloaded control plane. NFV is discussed at the ETSI NFV ISG and is introduced in [NFV-WHITEPAPER]. Mobile operator’s network is built
with variety of proprietary hardware appliances today. If we can get rid of these physical appliances and could shift to a cloud-based service, we will have a lot of benefits explained in the next section. This document assumes that NFV will push networking functions currently run on dedicated hardware onto a cloud network. Expected network functions are Mobility Management Entity (MME), Serving Gateway (SGW) PDN Gateway (PGW), etc. With NFV, EPC can be operated onto servers/hyper-visors. We name it virtualized-EPC (vEPC) in this document.

This document uses a lot of 3GPP specific terms. These terms can be found mostly at [RFC6459].

1.1. The Benefits of NFV

This section briefly explains the benefits of NFV. The detailed benefits can be found in [NFV-WHITEPAPER]. Although today’s eco-system of EPC appliances might be affected, we believe there are various approaches to enhance current eco-system and migrate to new NFV approaches. For example, operators could pay monthly recurring charges for the NFV services and operations to vendors, instead of one-time purchase and a little maintenance cost.

- **[Flexible Network Operations]**: The control functions of EPC are no longer in appliances deployed widely in operator’s network and can be run at hypervisor (cloud). It is easier to add and/ or delete functions from the services, because no physical construction is needed. Network operations will be much simpler and easier because complications of today’s network are pushed to NFV (i.e. hypervisor).

- **[Flexible Resource Managements]**: The EPC functions can be run on hypervisor and are now less dependent on proprietary hardware. Adding additional resources is easier in hypervisor, while adding or replacing physical appliances require installation, construction, configuration, and even migration plan without service cutoff. A hypervisor can be also shared across various functions such as PGW, SGW and MME. NFV also brings multi-tenancy and allows a single platform for different services and users. The operator can optimize resources and costs to share a NFV platform for multiple customers (ex. MVNO customers) and services (ex. multiple APNs).

- **[Faster Speed of Time to Market]**: When an operator wants a new function to its network and services, the operator needs to negotiate appliance vendors to implement the new functions or to find alternative equipment supporting the new function. It takes a longer time to convince the vendors, or to replace existing
hardware. However, if functions can be implemented as a software, it is much faster to implement the functions on NFV. Even the operator may implement them and try the new functions by themselves. Field trial is also getting easier because of no physical installation or replacement. You may turn on a new function in NFV and observe how the new function behaves in your network. NFV can save preparation time and tuning time of the new function.

o [Cost Optimization]: Last but not least, Cost is the most important motivation for operators to realize NFV. Operators can remove many of proprietary appliances from its network and replace them with industry standard servers, switches and routers. In addition, it is easy to scale up and down operator’s services so that resources can be always tuned to the size of services. In addition, operational costs led by any physical hardware such as power supply, maintenance, installation, construction and replacement can be minimized or even removed. The network design can be simpler, because complicated functions could be handled by NFV. That simple operation may enable automatic configurations and prevent unnecessary trouble-shooting. As a result, CAPEX and OPEX can be always optimized and lowered.

2. Motivations and Requirements, - Why IETF? -

2.1. Motivations

What is a role of IETF to realize vEPC in the future? IETF is not the right place to discuss, for instance, how to run MME on hypervisor. An important IETF activity must be to decouple the control- and user- planes of mobility protocols used in EPC. The motivation of decoupling the user and control plane is discussed in [I-D.wakikawa-req-mobile-cp-separation]. In doing so, NFV-enabled solutions can be easily designed and implemented with interoperability across multiple vendors and platforms. Otherwise, NFV solutions can be easily fragmented due to many proprietary solutions for the protocol separations. As stated in [NFV-WHITEPAPER], interoperability is highly important.

In the past, IETF has developed tunnel based mechanisms for mobile nodes such as Mobile IPv6 [RFC6275][RFC5555], Proxy Mobile IPv6 [RFC5213][RFC5844] and NEMO [RFC3963]. Similarly, 3GPP has developed tunnel protocols called GPRS Tunneling Protocol (GTP). These tunnel-based protocols establish a data path for a mobile node between the mobile node and an anchor point (s). There is a case where an access router terminates a tunnel instead of a mobile node (ex. Proxy Mobile IP). In 3GPP, a tunnel is established between SGW and PGW per a mobile node by either Proxy Mobile IPv6 or GTP. The control and
the user planes of these mobility protocols are tightly related and cannot be decoupled. The signaling like Binding Update and user’s packets are routed along a same path in EPC. It might be necessary to extend these mobility protocols for the user- and control-planes separation. The protocol separation of Mobile IP is discussed in [I-D.yokota-dmm-scenario].

Alternatively, if vEPC was realized, we should have an opportunity to re-visit the basic architecture of mobility system. Instead of tunneling packets on today’s EPC, why can’t we just route packets to a mobile node? Since a role of the user plane is "routing", BGP and other routing protocols could be used to forward UE’s traffic. This document introduces a BGP-based solution. Software Defined Networking (SDN) can be an alternative solution. Open Flow and other relevant protocols can setup the forward path dynamically according to UE’s states available in the control plane.

We have to remember that there is a good reason of adapting tunneling in Mobile IP based solutions, that is global mobility and signaling. A mobile node should be able to move anywhere on the Internet and be reachable from anyone on the Internet. There were routing based global mobility solutions like Boeing global mobility [Boeing-BGP] and WINMO [RFC6301]. In these proposals, BGP was used to propagate forwarding information of mobile nodes to the Internet. Whenever a mobile node changes its point of attachment, the route must be updated. Due to scalability and stability issues of the Internet, this solution was not recommended by IETF [Boeing-BGP]. However, as Boeing showed, it is doable to support global mobility by using BGP routing update. If scalability is not your concern, a routing based approach becomes a candidate of the mobility solution.

While global mobility is important, the "reality" is that your cell phones (i.e. UE/mobile node) are moving just within an operator’s network and fully controlled in your local EPC. If mobility is limited within an operator, we believe a routing based approach is feasible and practical for today’s mobile system. Instead of dedicated proprietary equipment like SGW and PGW to manage a tunnel path for a mobile node, multiple industry standard routers and switches are configured in the user plane. These switches and routers receive mobile nodes’ forwarding information from the control plane of vEPC by routing update.

2.2. Requirements

Requirements of our stateless user plane for vEPC are followings.

* NFV Support*
The future EPC architecture must support NFV capability. The control plane of EPC operated on NFV framework is named "virtualized EPC (vEPC)" in this document. The control plane of vEPC should keep backward compatibility with the today’s EPC’s control plane. It means this document doesn’t modify the control plane at all. It only assumes software-based MME, SGW, and PGW run on hypervisor.

Separation of Control- and User- Planes
Due to tight relationship of the control- and user- planes in today’s EPC, resource increase is always provisioned to both planes at once. It prevents flexible resource arrangement and introduces high capital investment and over-provisioned resources to one of planes. If NFV is deployed, it is expected that computing resources can be independently allocated to the control planes of the vEPC in a flexible manner.
Figure 1 shows a possibility that the entities of EPC Control-plane are virtualized in generic cloud environment, however user packets won’t go through those virtualized EPC nodes. Decoupling User-plane from the Control-plane entities will be made virtualized Control-plane nodes relax hypervisor data-path capacity requirements. On the other hand, decoupled User-plane into IP routing network will be agnostic from sessions and bearers states, of which are generated and maintained in the Control-plane. In terms of IP routing, forwarding packets through the networks is based on the destination address of the packets evaluated with network reachable information in the routing table that accommodated in the routing nodes. To forward EPC User-plane packets correctly, those states must be indicated by network reachable information.
Today’s 3GPP architecture introduces PDN gateway (PGW) as a gateway to external networks like the Internet. PGW manages all traffic from and to UEs and could be a bottleneck and single point of failure of network connectivity. In addition, due to recent rapid traffic increase, it is important to perform traffic engineering and to offload traffic to multiple locations (ex. SGW, PGW, eNodeB). For enhancements of traffic engineering capability, more flat design with multiple gateways is expected so that traffic can be distributed to all these gateways. There were proposals how to enable flat design to (Proxy) Mobile IP such as [I-D.wakikawa-mext-haha-interop2008] in IETF. Distributed Mobility Management (DMM) Working Group has also discussed how to extend Mobile IP-based solutions to support traffic distribution in an optimal way by removing centrally deployed anchors that is like a Home Agent.

Stateless in User Plane

Ultimate goal of vEPC is to remove all mobility specific states from the forwarding nodes in the user-plane of vEPC. If we succeed in this, industry standard routers and switches can be used to forward mobile nodes traffic in the user plane of vEPC. A mobile node’s specific states are kept in both an IP header of the mobile node’s packets and a routing entry of the mobile node. The detail is described in Section 3.2

3. Stateless user-plane architecture for virtualized EPC

This section explains our solution that is the stateless user-plane architecture for vEPC. This solution is basically a combination of existing protocols defined in IETF. A minor extension might be needed but it should be easily addressed in IETF. We first introduce our architecture and then protocol overview.

3.1. Architecture Overview

Figure 2 shows the user plane of the current EPC architecture. A tunnel is established between SGW and PGW by either Proxy Mobile IP or GTP. PGW is an anchor point of UE for incoming packets. All the packet destined to UE is routed first to PGW. The UE’s packets are intercepted by PGW and tunnelled to SGW. SGW then forwards the packet to UE via access points (i.e. eNodeB) over Radio Area Network (RAN).
User plane of the current EPC

Figure 2

Figure 3 is our proposed user plane of vEPC. The control plane is not shown in this figure.

User plane of vEPC

Figure 3

We introduce two new entities such as

EPC Edge Router (EPC-E)

EPC-E is located at the same place of today’s SGW and terminates GTP tunnel established with eNodeB (RAN). EPC-E supports the user plane functions of SGW and PGW. EPC-E is configured an anycast address to the network interface facing to eNodeB. The eNodeB establishes a GTP tunnel per UE with this anycast address. Thanks for anycast address, UE’s traffic forwarded by eNodeB is always routed to the closest EPC-E of UE. EPC-E is a router and
maintains routing information of every UE that is notified by the control plane. Detail of routing mechanism can be found in Section 3.4.

Router (RTR)
It is a regular IP router. The control plane of vEPC distributes routing information of every UE by a routing protocol like BGP. Therefore any additional protocols other than routing protocols are not needed for RTR. Multiple RTRs can be configured anywhere in the user plane of vEPC. RTRs announce UE’s routing information to the external network (ex. The Internet).

As you see in Figure 3, we omit a tunneling mechanism originally established between SGW and PGW for routing UE’s packets in the user plane. By removing this tunnel, UE’s packets are forwarded to and from the Internet according to routing tables on routers in the core network. Note that, although we remove the tunnel for UE’s traffic in the user plane, the control-plane signaling stays same in the control plane. If Proxy Mobile IP is used for this tunnel, Proxy Binding Update and Acknowledgment are exchanged between PGW and SGW that are managed by NFV on servers/hyper-visor. Instead of a tunnel setup, states created by Proxy Mobile IP are distributed to all routing entities (EPC-E and RTR) by a routing protocol. From the user plane point of view, these states are just seen as routing entries. EPC-E and RTR are not involved in any signaling of the control plane. The control plane just injects routing information to EPC-E and RTR to setup routing paths to and from UEs.

Although this architecture just uses IPv6 core network, it supports both IPv4 and IPv6 packets. The detailed operation of IPv4 support will be discussed in Section 3.5.

3.2. Protocol Overview

This section gives an example of protocols used for vEPC. Figure 4 is the procedure of the PDN connection setup in vEPC. This figure is copied from the section 3 of [RFC6459]. All the steps from (1) to (13) are same as the original except for NFV-based MME, SGW, PGW, HSS, and AAA.

The vEPC introduces two new steps, (14) and (15), to setup paths in the user-plane after finishing all the signaling on the control-plane. (16) and (17) are the steps to assign IP address to the mobile node.

In (14), vEPC advertises a routing information of UE to EPC-Es immediately right after the control-plane signaling completion. In this document, the advertising entity is a BGP speaker so that the
BGP speaker requires the control-plane to expose user-plane information for UEs.

To do that, the vEPC may utilize the Forwarding Policy Configuration Protocol (FPCP) [I-D.wt-dmm-fpc-cpdp] that defines FPCP Agent function and Client function. In FPCP enabled vEPC, the mobility control-plane should utilize FPCP Agent function to export that mobility state into the BGP speaker.

BGP speaker in the vEPC should utilize FPCP Client function to receive exported information from the agent. When the BGP speaker gets that information, they extract forwarding policies from them and generate BGP UPDATE messages that contain UE’s IP prefix in NLRI, endpoint information of tunnel, such as serving eNodeB, EPC-E address and its TEID, in a BGP attribute. BGP Remote Next-Hop attribute [I-D.vandevelde-idr-remote-next-hop] specifies the format to indicate GTP-U endpoint in the BGP attribute.

The EPC-E has peering with the BGP speakers directly. It is thus expected that there is no additional propagation delay of traversing multiple BGP speakers between EPC-E and vEPC. Adding that kind of surplus delay affects user-plane to be interrupted so that it should be avoided as much as possible for user experience.

In step (15), the EPC-E advertises routes to upstream routers such as the RTR. For scalable routing operation, UE’s prefixes should be aggregated into more shorter length prefixes. Due to that reason, the EPC-E generates routing information and advertised it to the RTR that includes aggregated prefix instead of UE’s prefixes and EPC-E address as the next-hop.
Extended PDN Connection Setup Procedure (copied Figure 8 of RFC6459)

Figure 4

UE requests an IPv6 prefix for its address assignment in the step (16). In our architecture, an IPv6 prefix is still assigned by vEPC in the control plane, as PDN-GW does in the legacy EPC. However,
EPC-E is responsible to deliver the IPv6 prefix to UE by DHCP or Stateless address autoconfiguration (SLAAC).

We now explain how EPC-E can know the prefix assigned to UE from vEPC for address configuration steps (16 and 17). When (1) to (15) are completed, vEPC has already advertised the UE’s prefix as route information to all the EPC-E. Therefore, when EPC-E receives a packet of either Router Solicitation or DHCPv6 request message, it just looks up the remote next-hop field of its routing information base (RIB) with the source IP address and the TEID of the received packet. A route entry matched for this search is the prefix delegated to the requesting UE. Therefore, EPC-E simply uses the prefix of the route entry as an assigned UE’s prefix.

In (17), EPC-E returns the found prefix to UE by either Router Advertisement or DHCPv6 reply message. UE now creates an address(es) from the received prefix. It is important to highlight that UE can obtain the same prefix information from any EPC-E all the time because the same UE’s route information is available on all the EPC-E.

It would be convenient to use automatic UE’s prefix creation rule or algorithm for vEPC. There are various mechanisms to create UE’s prefix. As an example, Stateless IPv6 Prefix Delegation [I-D.savolainen-stateless-pd] is introduced as an algorithm to create UE’s prefix in vEPC below. It important to mention that our architecture of the stateless user plane does not rely on any particular prefix creation mechanisms like [I-D.savolainen-stateless-pd] and can be run with any of them.

In the case of an UE’s prefix length is equal, or shorter than /64, the generated prefix is consisted as shown in Figure 5. Each PDN is assumed to have single or several prefixes (named PDN prefix) used to generate UE’s address. Followed by the PDN prefix, there is TEID field assigned for a UE’s session on S1-U interface of vEPC. TEID is 32 bits identifier in GTP header to distinguish each bearer. The remaining bits are filled by subnet ID.
3.2.1. Hand-over

When tunnel endpoint is updated by UE hand-over between eNodeBs, vEPC must refresh the route of UE with the updated tunnel endpoint as new remote next-hop.

Figure 6 shows vEPC that advertising updated route in (8) when UE hand-over from source eNodeB to target eNodeB on simplified hand-over procedure. The updated route that points to target eNodeB’s S1-U address and TEID as the next-hop should be immediately advertised to all the EPC-Es right after the procedures (1) to (7) completed.
Simplified Hand-over Procedure  

Figure 6

It is noted that RTR or any upstream routers of EPC-Es do not require routing update for each of UE hand-over event. EPC-E is required to just advertise once aggregate route during at least an UE route exist so that EPC-E does not advertise hand-over UE route in Figure 6. Operators require that their core network must be kept its routing stable. This architecture prevents routing fluctuation in the network that helps to fulfill that requirement consequently.

3.2.2. Detaching UE

In the case of UE detachment, vEPC also advertises route update that includes detached UE prefix as withdrawn route to delete the route of the detached UE from EPC-Es.
3.3. Control-plane awareness of stateless user-plane

Nodes in the control-plane in vEPC must be aware that the anycast address assigned to EPC-E is a S1-U address of vEPC. The vEPC must use the anycast address in signaling between vEPC and RAN. By doing this, packets from RAN are correctly forwarded to an appropriate EPC-E. Due to anycast nature, it means there is no hand-off procedure between SGWs because all eNodeB in the RAN send packets to the same anycast address.

When an operator needs to increase virtualized instances to cope with just signaling overload, the operator should use the existing S1-U address (i.e. EPC-E anycast address) for the new instances. If the operator would increase the capacity of the user plane, it can add additional EPC-Es in the core network. The operator can group the new EPC-Es as a set and increase scalability and performance of the user plane. In this case, the operator uses a new anycast address to the new set of EPC-E. We will discuss operational consideration in detail in Section 4.

3.4. Routing mechanism

Figure 7 shows a packet forwarding mechanism of our stateless user plane. As an example, there are four eNodeB (illustrated as eNB-x), three EPC-Edge routers (shown as EPC-Ex) and two routers (RTRx) in Figure 7. UE is first connected to eNB-C and then moves to eNB-D. The UE at the new location is illustrated as UE’. Routing entry for UE is also illustrated at the right side in Figure 7.

EPC-E has two interfaces facing either RAN or CORE networks. An anycast address (shown as X) is configured to the interface facing RAN of all EPC-E. EPC-E assigns an individual IPv6 address to another interface (illustrated "a" to "d" in the figure). It is important to mention that the anycast address X can be treated as the SGW’s S1-U address.

Since RTRs are a gateway to the Internet, they advertise routes of an operator’s prefix to the Internet. After one of RTR receives a packet of UE from the Internet, it needs to routing it to UE in the user plane. RTR has a simple routing entry for PDN prefix whose next hop points to the EPC-E. One of RTR (let’s say RTR2 in this case) looks up a routing table with UE’s address and matched it with a routing entry of PDN prefix. Since multiple EPC-Es advertise a route for the same PDN prefix, RTR2 should forward the packet to one of EPC-E according to the routing entry. This routing is known as hot-potato routing. In this example, the RTR2 uses EPC-E2-b as a nexthop of PDN prefix.
When the UE’s packet is arrived at EPC-E2, EPC-E2 needs to forwards them to UE via eNodeB to which UE is connecting by using GTP tunnel. For this operation, EPC-E2 has a routing entry that destination is UE’s prefix and that next hop points to GTP tunnel between eNB-C and the EPC-Es. In order to identify the GTP tunnel for UE, EPC-E needs S1-U address and Tunnel Endpoint ID (TEID) of eNB-C that is eNB-C-3 in Figure 7. The eNB-C TEID for UE is illustrated as TEID[eNB-C]. The SGW assigned TEID is utilized to generate the UE’s prefix as we explained in Section 3.2. These TEID are assigned per UE. The TEID and S1-U address of eNodeB are retrieved from the next hop field of the routing entry of the mobile node. By using the GTP information, every EPC-E can now forward the UE’s packets to right eNodeB.

Routing outgoing packets from UE is much simpler. The packets from UE are always routed to the closest EPC-E to UE because of anycast routing. In Figure 7, when UE sends a packet to a destination, the packet is reached to eNB-C and tunneled to EPC-E’s anycast address. The GTP-tunneled packet is routed to the closest EPC-E that is EPC-E2 in this case. The packet is decapsulated by EPC-E2 and then forwarded to one of RTR according to the routing table. Since the decapsulated packet is regular IPv6 packet, no extra control other than routing is necessary.

When UE moves to a new location (UE’), it updates its location on the control plane. After signaling completion for location update, vEPC needs to update the UE’s routing entry of all EPC-E so that vEPC advertises updated route with new location to all EPC-Es by a routing protocol. The routing entry should be updated with the new eNodeB’s address that is eNB-D-4. During handover, there might be some traffic arriving to the older eNodeB (eNB-C). These packets can be re-routed to the new eNodeB (eNB-D) via X2-U interface in RAN.

The UE’s address isn’t changed when UE changes its attachment. In our scenario, SGW run on hypervisor and is independent from network topology. Therefore, logically we don’t have handover across different SGWs. UE can stay connected with the same SGW all the time and can keep using the same TEID after handover. Thus, UE’s address is unchanged even after handover.
Routing Mechanism Overview

Figure 7

*1 TEID used at EPC-E for the UE is included in this UE’s prefix. see Figure 4.

*2 GTP tunnel state is stored in the next hop field. The state information is the combination of eNB-C S1 address that is eNB-C-3 and TEID(eNB-C) assigned for the UE.
3.5. IPv4 Support

Recent IPv6 transition mechanisms enable IPv6-only network to forward IPv4 packet with encapsulation or translation techniques. By using one of mechanisms, we can use IPv6 for our stateless user-plane network for transporting both IPv4 and IPv6 packets. Figure 8 shows available solutions of IPv4 support for each bearer type to deal with that requirement.

<table>
<thead>
<tr>
<th>Bearer type</th>
<th>UE function</th>
<th>EPC-E function</th>
<th>Gateway function</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4</td>
<td>-</td>
<td>B4</td>
<td>AFTR</td>
</tr>
<tr>
<td>IPv4</td>
<td>-</td>
<td>CLAT</td>
<td>PLAT</td>
</tr>
<tr>
<td>IPv6</td>
<td>MAP-CE</td>
<td>-</td>
<td>MAP-BR</td>
</tr>
<tr>
<td>IPv6</td>
<td>B4</td>
<td>-</td>
<td>AFTR</td>
</tr>
<tr>
<td>IPv6</td>
<td>CLAT</td>
<td>-</td>
<td>PLAT</td>
</tr>
</tbody>
</table>

Solutions and functions for IPv4 support

Figure 8

In the case of a UE only support IPv4 bearer, B4 function of DS-Lite [RFC6333] or CLAT function of 464XLAT [RFC6877] may be implemented in a EPC-E. Both functions are stateless therefore EPC-E isn’t required to maintain any tunneling or translation state.

Figure 9 shows how to support IPv4 on IPv6 core network in our vEPC. Instead of using RqTR as a gateway to the Internet, DS-LITE AFTR or 464XLAT PLAT is installed as a gateway to the IPv4 Internet.
IPv4 User plane of vEPC

Figure 9

If UE supports IPv6 capable bearer, IPv6 transition function may be implemented in the UE such as MAP-CE [I-D.ietf-softwire-map], B4 or CLAT. That means an EPC-E receives IPv6 packets from UE in this case so that the EPC-E does not need to be involved in the part of IPv4 support functions.

4. Operational Considerations

4.1. Scalability and Reliability

Virtualization allows vEPC to be elastic for steep demand of requests to create and update for sessions. In our architecture, that makes routing update fluctuation from vEPC to EPC-E. This is the reason why we select BGP as a protocol between vEPC and EPC-E. BGP is scalable and stable routing protocol today.

BGP is an incremental update protocol so that once BGP peer established, millions of routes can be easily updated in stable manners. Operators can appropriately design BGP peering between vEPC and EPC-E to secure convergence time within appropriate period.

Granularity of the peering should be aware EPC-E capacity because it is assumed that EPC-E has upper limit of routing entries. BGP peering design should makes sure that total number of routes does not exceed EPC-E capacity.

During the network planning, operators must understand EPC-E’s capacity such as # of routes, bandwidth, etc. An example of estimation, if a EPC-E has 1Gbps throughput and each UE’s bandwidth consumption is 10Kbps in average, the EPC-E should have 100K routes capacity.
This is an operational approach to minimize the risk of routing update fluctuation. If it is hard to support all the UEs by a EPC-E in an operators network, another EPC-E can be introduced and configured as a set of EPC-Es. The UEs are distributed and handled by the EPC-Es within the set. We don’t need to support millions of UEs by a single EPC-E.

EPC-E set is also useful to have EPC-E redundancy for reliable operation. The nature of BGP makes easy to replicate UE routes to multiple EPC-Es within a EPC-E set. In that EPC-E set, when an EPC-E fall down to a failure, another EPC-E come out with same UE routes that the fall-down EPC had and immediately re-converge to core routing. That helps user-plane to minimize disruption during EPC-E failure recovery.

These are another advantage of using routing mechanism in the user plane. We already explain how to handle multiple EPC-Es and EPC-E sets in our scheme in Section 3.3.

The notion of multiple EPC-E sets is easily fitted into our today’s’ network. The operator’s network is often separated into several regional network for geographical scalability. Therefore, the operator can assign different EPC-E set to different region for better scalability.

In that network, when an UE hands over between two regions, the session of the UE might be disconnected if the serving EPC-E doesn’t have reachability for those region access networks. For example, in the case of regional access networks have duplicated IPv4 private address space. To enable inter-region hand-over, it is recommended that all of the access network, such as RAN, are IPv6 networks and reachable each other.

In addition, routers and EPC-E in the IPv6 core network are required to process just "route", they naturally aggregate those routing entries. It helps limiting the total number of routing entries in our core network.

4.2. Backward Compatibility

vEPC should be able to fall back to the legacy EPC based packet forwarding to secure backward compatibility which is required to connect existing system, or to connect roaming partners through legacy S5/S8 interfaces. When fallback happened, all the packets are not routed on our stateless user plane, but forwarded to vEPC (i.e. SGW and PGW instances on hypervisor). vEPC must use a S1-U address that is different from anycast address assigned to EPC-Es. This
address is assigned to SGW instances in vEPC and used to terminate
tunnels in vEPC servers (i.e. hypervisor).

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

There are no security considerations specific to this document at
this moment.

7. References

7.1. Normative References

[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-08 (work in progress), October 2014.

Subsequent Address Family Identifier (SAFI) and the BGP
Tunnel Encapsulation Attribute", RFC 5512, April 2009.

7.2. Informative References

[Boeing-BGP]
Andrew, , "Global IP Network Mobility using Border Gateway
Protocol (BGP)", IAB Plenary IAB Plenary of IETF 62nd,
March 2005.

[I-D.ietf-softwire-map]
Troan, O., Dec, W., Li, X., Bao, C., Matsushima, S.,
Murakami, T., and T. Taylor, "Mapping of Address and Port
with Encapsulation (MAP)", draft-ietf-softwire-map-13
(work in progress), March 2015.

[I-D.savolainen-stateless-pd]
Savolainen, T. and J. Korhonen, "Stateless IPv6 Prefix
Delegation for IPv6 enabled networks", draft-savolainen-
stateless-pd-01 (work in progress), February 2010.

[I-D.wakikawa-mext-haha-interop2008]
Wakikawa, R., Shima, K., and N. Shigechika, "The Global
HAHA Operation at the Interop Tokyo 2008", draft-wakikawa-
mext-haha-interop2008-00 (work in progress), July 2008.
[I-D.wakikawa-req-mobile-cp-separation]
Wakikawa, R., Matsushima, S., Patil, B., Chen, B., DJ, D.,
and H. Deng, "Requirements and use cases for separating
control and user planes in mobile network architectures",
draft-wakikawa-req-mobile-cp-separation-00 (work in
progress), November 2013.

[I-D.wt-dmm-fpc-cpdp]
Liebsch, M., Matsushima, S., Gundavelli, S., and D. Moses,
"Protocol for Forwarding Policy Configuration (FPC) in
DMM", draft-wt-dmm-fpc-cpdp (work in progress), March
2015.

[I-D.yokota-dmm-scenario]
Yokota, H., Seite, P., Demaria, E., and Z. Cao, "Use case
scenarios for Distributed Mobility Management",
draft-yokota-dmm-scenario-00 (work in progress), October
2010.

[NFV-WHITEPAPER]
Network Operators, , "Network Functions Virtualization, An
Introduction, Benefits, Enablers, Challenges and Call for
Action", SDN and OpenFlow SDN and OpenFlow World Congress,
October 2012.

Thubert, "Network Mobility (NEMO) Basic Support Protocol",

[RFC5213] Gundavelli, S., Leung, K., Devarapalli, V., Chowdhury, K.,

[RFC5555] Soliman, H., "Mobile IPv6 Support for Dual Stack Hosts and
Routers", RFC 5555, June 2009.


[RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support

Support in the Internet", RFC 6301, July 2011.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-
Stack Lite Broadband Deployments Following IPv4
Exhaustion", RFC 6333, August 2011.


Authors’ Addresses

Satoru Matsushima
Softbank Telecom
1-9-1,Higashi-Shimbashi,Minato-Ku
Tokyo 105-7322
Japan

Email: satoru.matsushima@g.softbank.co.jp

Ryuji Wakikawa
Softbank Mobile
1-9-1,Higashi-Shimbashi,Minato-Ku
Tokyo 105-7322
Japan

Email: ryuji.wakikawa@gmail.com
DHCPv6 Extension for On Demand Mobility exposure
draft-moses-dmm-dhcp-ondemand-mobility-00

Abstract

Applications differ with respect to whether they need IP session
continuity and/or IP address reachability. Networks providing the
same type of service to any mobile host and any application running
on the host yields inefficiencies. This document describes
extensions to the DHCPv6 protocol to enable mobile hosts to indicate
the required mobility type of the requested IP address, and networks
to indicate the type of mobility service associated with the
allocated IP address in return.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 9, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents
(http://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
1. Introduction

[TBD reference to the On-demand draft] defines different types of mobility-associated services provided by access networks to mobile hosts with regards to maintaining IPv6 address continuity after an event of the host moving to different locations with different points of attachments within the IP network topology. It further specifies means for applications to convey to the IP stack in the mobile host, their requirements regarding these services.

This document specifies extensions to the DHCPv6 protocol specified in [RFC3315] in the form of a new DHCP option that specifies the type of mobility services associated with an IPv6 address. The IP stack in a mobile host uses the DHCP client to specify the type of mobility service to be associated with an expected source IPv6 address. The network uses the DHCP server to convey the type of service it is committed to provide with the assigned IPv6 address using this option.

The type of service is associated by the network with the source IPv6 address assigned to the mobile host. For example, if a mobile host requests IP address continuity throughout the life of the IP session and the network commits to provide that service, it will associate the service with an assigned source IPv6 address, and reply with the IPv6 address and an indication of the type of service associated with that address.
2. Notational Conventions

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

3. IPv6 Continuity Service Option

The IPv6 Continuity Service option is used to specify the type of continuity service associated with a source IPv6 address. The IPv6 Continuity Service option must be encapsulated in the IAaddr-options field of the IA Address option.

The format of the IPv6 Continuity Service options is:

TBD - Add format description...

In a message sent from a client to a server, the value of the IPv6 Continuity Service option indicates the type of IP continuity required for the IPv6 address requested by the client.

In a message sent from a server to a client, the value of the IPv6 Continuity Service option indicates the type of IP continuity service committed by the network for the associated IPv6 address.

If a server received a request to assign an IPv6 address with a specified IPv6 Continuity service, but cannot fulfill the request, it must reply with the [TBD] status.

A server that does not support this option will discard it as well as the IA Address option that had this option encapsulated in one of its IAaddr-options field.

If a client does not receive the requested address, it must resent the request without the desired IPv6 Continuity Service option since it is not supported by the server. In that case, the host of the client cannot assume any IP continuity service behavior for that address.

A server must not include the IPv6 Continuity Service option in the IAaddr-options field of an IA Address option, if not specifically requested previously by the client to which it is sending a message.

If a client receives an IA Address option from a server with the IPv6 Continuity Service option in the IAaddr-options field, without initially requesting a specific service using this option, it must discard the received IPv6 address.
If the mobile host has no preference regarding the type of continuity service it uses the ‘AnyType’ value as the specified type of continuity service. The Server will allocate an IP address with some continuity service and must specify the type in IPv6 Continuity Service option encapsulated in the IAaddr-options field of the IA Address option. The method for selecting the type of continuity service is outside the scope of this specification.

4. Security Considerations

There are no specific security considerations for this option.

5. IANA Considerations

TBD

6. References

6.1. Normative References


6.2. Informative References


Authors' Addresses

Danny Moses
Intel
Petah Tikva
Israel

Email: danny.moses@intel.com
Alper Yegin
Samsung
Istanbul
Turkey

Email: alper.yegin@partner.samsung.com
Abstract

This draft specifies and analyzes the expected cases regarding the selection of a proper source IP address and address type based on the application features over a distributed mobility management (DMM) network. It also provides available selection methods in the specified scenarios.

Status of this Memo

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

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

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

This Internet-Draft will expire on August 26, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
1. Introduction

In [draft-yegin-dmm-ondemand-mobility], it makes an attempt to classify the source IP address type for a mobile host, depending on the need of IP session continuity and/or IP address reachability. Therefore, three types of IP addresses were defined with regard to the mobility management; fixed IP address, sustained IP address, and nomadic IP address.

After introducing the three types of IP addresses, it proposed a solution for the applications running on the mobile host to indicate whether they need IP session continuity or IP address reachability.

When an application tries to get an IP address, it may require or prefer specific type of IP address or non-specific (any) type of it to the IP stack. The proposed approach aims to obtain a proper IP address.
address corresponding to a specific address requirement, whereas the former approaches [RFC5014][RFC6724] operate on the available set of IP addresses, based on a preference.

But even in the specific type of IP address request, there may be a need to indicate further requirements such as which IP address is more preferred among already configured multiple IP addresses based on the same type requested. Such a situation is easily met over a DMM network environment for some reasons such as QoS or Policy, as a mobile host is supposed to obtain a new prefix at each new mobility access router.

Aligned with the needs, this draft specifies and describes expected use cases and proposes required extensions to support the given use cases. This draft is based on the [draft-yegin-dmm-ondemand-mobility] that proposed the three types of IP addresses with regard to the mobility management.

2. Use Cases

We specify and analyse expected use cases when an application session is initiated. Furthermore, we organize the use cases according to the requested IP address type and additional requirement.

2.1. When an application does not need to request a specific IP address type and requirement

Applications such as a text-based web browsing or information-centric service, e.g. weather and stock information may belong to this category. The suggested flag, IPV6_REQ_NOMADIC_IP, defined in [draft-yegin-dmm-ondemand-mobility] is used for expressing its preference to the IP stack. But it does not require a further signaling between the mobile host and the network, as a nomadic IP address is obtained by default whenever the mobile host is attached at an (mobility) access router. That is, obtaining this type of IP address could be orthogonal with the IP request by the application. However, it is only valid while the mobile host stays at the attached mobility access router.

2.2. When an application needs to request specific IP address type and requirement

This category is for an application requiring IP session continuity with different granularity of IP address reachability. This case is again divided by three sub-cases with regard to IP address type availability and/or address selection, if needed. But the request of
nomadic IP address is excluded in following cases, since it is given by default as described in Section 3.1.

2.2.1. Case 1: there is no available IP address based on a requested type in the IP stack.

For resource-efficiency mobility support, the dynamic configuration of a sustained IP or fixed IP address can be preferred. Since there is a nomadic IP address configured in the IP stack, when a new type of IP address is needed, additional support mechanism is needed to express the preference of the application.

In this case, the IP stack triggers one of the IP address configuration mechanisms (e.g. DHCPv6, SLAAC, or IP mobility protocols).

2.2.2. Case 2: there are one or more IP addresses based on a requested type in the IP stack, and no selection preference by the application.

The mobile host already has the IP addresses following a requested IP address type by the application. In this case, the default address selection rules will apply [RFC6724], i.e. scope preference and longest prefix matching. The best-matched IP address among them will be selected.

2.2.3. Case 3: there are one or more IP addresses based on a requested type in the IP stack, but there is a selection preference by the application.

In case of fixed IP address, the default address selection rule can simply be applied so that one IP address can be selected.

In case of sustained IP address, taking into account the benefits of on-demand mobility from several DMM solution proposals, it is highly preferred for a mobile host to use a sustained IP address based on the prefix from a currently attached router.

By following the default address selection algorithm, only the best-matched IP address will be selected, which may not be "the best" in terms of optimal routing and network resource spent. Indicating the host's preference will be required (See Section 4 for the proposed flag).

For instance, suppose that an MN has already a sustained IP address (PrefA::) obtained in the IP stack when it stayed at network A and now it moved to network B. We also suppose that another application
wants to configure a sustained IP address, but based on a new prefix from currently attached network for optimal routing. Without any indication by the application, the existing sustained IP address (PrefA::) will be selected and the session will be anchored at network A, which may lead to inefficiency to application performance and network resource due to sub-optimal routing.

3. Indications for expressing requirements

When an application prefers a new IP address of the requested IP address type, additional indication flags should be delivered through the socket API interface.

3.1. Suggested indication flag

IPV6_PREFER_SRC_NEW

/* Prefer a new IP address based on a requested IP address type as source */

This flag is proposed to be added in RFC5014, and aims to express the preference for enabling differentiated per-flow anchoring. The use of the flag can be combined together with the three types of IP address defined in [draft-yegin-dmm-ondemand-mobility]. It is in equal degree and orthogonal with the defined flag-set in IPv6 socket API for source address selection [RFC5014].

4. Security Considerations

T.B.D.

5. IANA Considerations

T.B.D.

6. References

6.1. Normative References


Authors’ Addresses

Seil Jeon
Instituto de Telecomunicacoes
Campus Universitario de Santiago
Aveiro 3810-193, Portugal
E-mail: seiljeon@av.it.pt

Sergio Figueiredo
Universidade de Aveiro
3810-193 Aveiro, Portugal
E-mail: sfigueiredo@av.it.pt

Younghan Kim
Soongsil University
369, Sangdo-ro, Dongjak-gu,
Seoul 156-743, Korea
E-mail: younghak@ssu.ac.kr
Transmission of IP Packets over AERO Links
draft-templin-aerolink-52.txt

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on August 14, 2015.
Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
2. Terminology .................................................. 4
3. Asymmetric Extended Route Optimization (AERO) .................. 6
   3.1. AERO Link Reference Model .............................. 6
   3.2. AERO Link Node Types ................................. 8
   3.3. AERO Addresses ...................................... 9
   3.4. AERO Interface Characteristics ...................... 10
   3.5. AERO Link Registration ............................... 11
   3.6. AERO Interface Initialization ....................... 12
      3.6.1. AERO Relay Behavior ........................... 12
      3.6.2. AERO Server Behavior .......................... 12
      3.6.3. AERO Client Behavior .......................... 12
      3.6.4. AERO Forwarding Agent Behavior ............... 13
   3.7. AERO Link Routing System ............................. 13
   3.8. AERO Interface Neighbor Cache Maintenance ............... 14
   3.9. AERO Interface Sending Algorithm ..................... 16
   3.10. AERO Interface Encapsulation and Re-encapsulation ...... 17
   3.11. AERO Interface Decapsulation ....................... 20
   3.12. AERO Interface Data Origin Authentication .............. 20
   3.13. AERO Interface MTU and Fragmentation ................. 21
      3.13.1. Accommodating Large Control Messages .......... 23
      3.13.2. Integrity .................................. 24
   3.14. AERO Interface Error Handling ...................... 25
   3.15. AERO Router Discovery, Prefix Delegation and Address Configuration ............................ 29
      3.15.1. AERO DHCPv6 Service Model .................... 29
      3.15.2. AERO Client Behavior .......................... 30
      3.15.3. AERO Server Behavior ......................... 32
   3.16. AERO Forwarding Agent Discovery ..................... 36
   3.17. AERO Intradomain Route Optimization .................. 36
      3.17.1. Reference Operational Scenario ................ 37
This document specifies the operation of IP over tunnel virtual links using Asymmetric Extended Route Optimization (AERO). The AERO link can be used for tunneling to neighboring nodes over either IPv6 or IPv4 networks, i.e., AERO views the IPv6 and IPv4 networks as equivalent links for tunneling. Nodes attached to AERO links can exchange packets via trusted intermediate routers that provide forwarding services to reach off-link destinations and redirection services for route optimization that addresses the requirements outlined in [RFC5522].

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

2. Terminology

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

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

AERO interface
a node’s attachment to an AERO link. 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 then coordinates with other AERO nodes using IPv6 ND messaging.

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 assigned IPv6 link-local unicast address to support the operation of DHCPv6 and the IPv6 ND protocol.

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

AERO Forwarding Agent ("Forwarding Agent")
a node that performs data plane forwarding services as a companion to other AERO nodes.

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].

3. Asymmetric Extended Route Optimization (AERO)

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

3.1. AERO Link Reference Model
Figure 1 presents the AERO link reference model. In this model:

- 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.
- Servers S1 and S2 associate with Relay R1 and also act as default routers for their associated Clients C1 and C2.
- Clients C1 and C2 associate with Servers S1 and S2, respectively, and also act as default routers for their associated EUNs.
- 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 ACPs H1 and H2.) 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 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 unless they are also configured as a Relay. AERO Servers maintain an AERO interface neighbor cache entry for each AERO Relay. They also maintain both a neighbor cache entry and an IP forwarding table entry for each of their associated Clients.

AERO Clients act as requesting routers to receive ACPs through DHCPv6 PD exchanges with AERO Servers over the AERO link 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 other AERO nodes. Note that while all Relays, Servers and Clients 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 performance-intensive 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:

```
+----------------------------------+
| Type = 2 | Length = 3 | Reserved |
| Link ID  | Preference | 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, and Preference is an integer value between 0 and 255 indicating the
node's preference for this underlying interface (with 255 being the highest preference, 1 being the lowest, and 0 meaning "link disabled"). UDP Port Number and IP Address are set to the addresses used by the target node when it sends encapsulated packets over the underlying interface. When 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., "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 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 "example.com", the DNS database contains:

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

This DNS registration advertises the AERO link's ASPs to prospective mobile nodes.
3.6. AERO Interface Initialization

3.6.1. AERO Relay Behavior

When a Relay enables an AERO interface, it first assigns an administratively-assigned 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 the set of ASPs 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 assigned 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 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 entries 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 node that manages AERO control messaging services. The Forwarding Agent thereafter provides data plane forwarding services based solely on the forwarding information assigned to it by the companion AERO node. AERO Forwarding Agents perform NS/NA messaging, i.e., the same as for any AERO node.

3.7. AERO Link Routing System

Relays require full topology knowledge of all ACP/Server associations, 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) (possibly using a private AS Number (ASN) [RFC1930]), and each Server further peers with each Relay but does not peer with other...
Servers. Similarly, Relays 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.

In some environments, Relays need not send BGP updates to Servers since Servers can always use Relays as default routers, however this presents a data/control plane performance tradeoff. In environments where sustained packet forwarding over Relays is undesirable, Relays can instead report ACPs to Servers while including a BGP Remote-Next-Hop [I-D.vandevelde-idr-remote-next-hop]. The Server then creates a neighbor cache entry for each ACP with the Remote-Next-Hop as the link-layer address to enable Server-to-Server route optimization.

3.8. AERO Interface Neighbor Cache Maintenance

Each AERO interface maintains a conceptual neighbor cache that includes an entry for each neighbor it communicates with on the AERO link, the same as for any IPv6 interface [RFC4861]. AERO interface neighbor cache entries 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 short 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 neighbor, it may receive RA messages with one or more SLLAOs that include the link-layer addresses of AERO Forwarding Agents. The AERO node SHOULD record the link-layer addresses in the neighbor cache entry for the neighbor and send subsequent data packets via one of these addresses instead of the neighbor’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 base tunneling specifications (e.g., [RFC2003], [RFC2473], [RFC2784], [RFC4213], [RFC4301], [RFC5246], etc.) except that it inserts a UDP header immediately following the IP encapsulation header. If there are no additional encapsulation headers (and no fragmentation, identification, checksum or signature is needed), the AERO interface next encapsulates the IPv4 or IPv6 packet immediately following the UDP header. In that case, the most significant four bits of the encapsulated packet encode the value ‘4’ for IPv4 or ‘6’ for IPv6.

For all other encapsulations, the AERO interface MUST insert an AERO Header between the UDP header and the next encapsulation header as shown in Figure 3:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>+-----------------------------------------------+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Version</td>
<td>N</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>+-----------------------------------------------+</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+-----------------------------------------------+</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identification (32 bits)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+-----------------------------------------------+</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checksum (16 bits)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+-----------------------------------------------+</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Signature (variable length)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: AERO Header

**Version** a 4-bit "Version" field. MUST be 0 for the purpose of this specification.

**N** a 1-bit "Next Header" flag. MUST be 1 for the purpose of this specification to indicate that "Next Header" field is present. "Next Header" encodes the IP protocol number corresponding to the next header in the encapsulation immediately following the AERO header. For example, "Next Header" encodes the value ‘4’ for IPv4, ‘17’ for UDP, ‘41’ for IPv6, ‘47’ for GRE, ‘50’ for ESP, ‘51’ for AH, etc.

**F** a 1-bit "Fragment Header" flag. Set to ‘1’ if the "Fragment Offset", "Res", "M", and "Identification" fields are present and collectively referred to as the "AERO Fragment Header"; otherwise, set to ‘0’.

**C** a 1-bit "Checksum" flag. Set to ‘1’ if the "Checksum" field is present; otherwise, set to ‘0’. When present, the Checksum field contains a checksum of the IP/UDP/AERO encapsulation headers prior to the Checksum field.
S a 1-bit "Signature" flag. Set to '1' if the "Signature" field is present; otherwise, set to '0'. When present, the Signature field contains a cryptographic signature of the encapsulated packet following the Signature field. The signature is applied prior to any fragmentation; hence' the Signature field only appears in the first fragment of a fragmented packet.

(Note: [RFC6706] defines an experimental use in which the bits corresponding to (Version, N, F, C, S) are all zero, which can be unambiguously distinguished from the values permitted by this specification.)

During encapsulation, the AERO interface copies the "TTL/Hop Limit", "Type of Service/Traffic Class" [RFC2983] and "Congestion Experienced" [RFC3166] values in the packet’s IP header into the corresponding fields in the encapsulation IP header. (When IPv6 is used as the encapsulation protocol, the interface also sets the Flow Label value in the encapsulation header per [RFC6438].) 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 AERO 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 to zero (see: [RFC6935][RFC6936]) unless an integrity check is required (see: Section 3.13.2).

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. The AERO interface finally sets the AERO header fields as described in Figure 3.
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. When the AERO interface receives a UDP packet, it examines the first octet of the encapsulated packet.

If the most significant four bits of the first octet encode the value ‘4’ (i.e., the IP version number value for IPv4) or the value ‘6’ (i.e., the IP version number value for IPv6), the AERO interface discards the encapsulation headers and accepts the encapsulated packet as an ordinary IPv6 or IPv4 data packet, respectively. If the most significant four bits encode the value ‘0’, however, the AERO interface processes the packet according to the appropriate AERO Header fields as specified in Figure 3.

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:

- AERO Relays and Servers accept encapsulated packets with a link-layer source address that matches a permanent neighbor cache entry.
- 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.
- 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.
- 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.
- 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 (termed here "ENCAPS"), since the
maximum packet size in the base IP specifications is 64KB
[RFC0791][RFC2460] (while IPv6 jumbograms can be up to 4GB, they are
considered optional for IPv6 nodes [RFC2675][RFC6434]).

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. Additionally,
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 MTU discovery mechanisms
[RFC1191][RFC1981].

For these reasons, AERO interfaces admit all packets up to 1500 bytes
in length even if some fragmentation is necessary, and admit larger
packets without fragmentation in case they are able to traverse the
tunnel in one piece. AERO interfaces are therefore considered to
have an indefinite MTU, i.e., instead of clamping the MTU to a finite
size.

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 3.13.2). 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 it is performed through insertion of an
AERO fragment header (see: Section 3.10) and application of tunnel
fragmentation as described in Section 3.1.7 of [RFC2764]. Since the
AERO fragment header reduces the room available for packet data, but
the original source has no way to control its insertion, the header
length MUST be included in the ENCAPS length even for packets in
which the header does not appear.
The source AERO interface (i.e., the tunnel ingress) therefore sends encapsulated packets to the destination AERO interface (i.e., the tunnel egress) according to the following algorithm:

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

- For IP packets that are larger than \((1280-\text{ENCAPS})\) bytes but no larger than 1500 bytes, the tunnel ingress encapsulates the packet and inserts an AERO fragment header. Next, the tunnel ingress uses the fragmentation algorithm in [RFC2460] to break the packet into two non-overlapping fragments where the first fragment (including \(\text{ENCAPS}\)) is no larger than 1024 bytes and the second is no larger than the first. Each fragment consists of identical UDP/IP encapsulation headers, followed by the AERO header followed by the fragment of the encapsulated packet itself. The tunnel 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 tunnel egress. When the tunnel egress receives the fragments, it reassembles them into a whole packet per the reassembly algorithm in [RFC2460]. The tunnel egress therefore MUST be capable of reassembling packets up to 1500+\(\text{ENCAPS}\) bytes in length; hence, it is RECOMMENDED that the tunnel egress be capable of reassembling at least 2KB.

- For IPv4 packets that are larger than 1500 bytes and with the DF bit set to 0, the tunnel 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-\(\text{ENCAPS}\) and all other fragments are no larger than the first fragment. The tunnel 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 tunnel egress.

- For all other IP packets, if the packet is too large to enter the underlying interface following encapsulation, the tunnel 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 \(\text{ENCAPS}\). Otherwise, the tunnel 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 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 be either 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 tunnel ingress MAY also send 1500 byte or larger probe packets to determine whether they can reach the tunnel egress without fragmentation. If the probes succeed, the tunnel 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 tunnel 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 tunnel ingress prepares 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 tunnel ingress then encapsulates the padded NS message in the encapsulation headers (and for IPv4 sets DF to 1) then sends the message to the tunnel egress. If the tunnel egress returns a solicited NA message with a matching Nonce option, the tunnel ingress deems the probe successful. Note that in this process it is essential that probes follow equivalent paths to those used to convey actual data packets. This means that Equal Cost MultiPath (ECMP) and Link Aggregation Gateway (LAG) equipment is assumed to support identical MTUs along all paths.

3.13.1. 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:

- For control messages that are larger than (1280-ENCAPS) bytes, the tunnel ingress encapsulates the packet and inserts an AERO fragment header. Next, the tunnel ingress uses the fragmentation algorithm in [RFC2460] 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 tunnel 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 tunnel egress SHOULD be able to reassemble them if they do. This means that the tunnel 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 tunnel ingress can send large control messages without fragmentation if there is assurance that large packets can traverse the tunnel without fragmentation. The tunnel ingress MAY send 1500 byte or larger probe packets as specified above to determine a size for which fragmentation can be avoided.

3.13.2. 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 misassociation 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 AERO fragmentation procedure described in the above algorithms reuses standard IPv6 fragmentation and reassembly code. Since the AERO 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 tunnel 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 the AERO fragmentation and reassembly procedures.

When the tunnel 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 tunnel ingress would only need to produce $\sim(5 \times 10^6)$ 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 tunnel ingress MAY omit an integrity check for the IPv4 fragmentation and reassembly procedures; otherwise, the tunnel ingress SHOULD include an integrity check. When an upper layer encapsulation (e.g., IPsec) already includes an integrity check, the tunnel ingress need not include an additional check. Otherwise, the tunnel 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 tunnel 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 4:

```
+-----------------------------------+
<p>|        L2 IP Header of            |</p>
<table>
<thead>
<tr>
<th>error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2 ICMP Header</td>
</tr>
<tr>
<td>-----------------------------------</td>
</tr>
<tr>
<td>IP and other encapsulation</td>
</tr>
<tr>
<td>headers of original L3</td>
</tr>
<tr>
<td>packet</td>
</tr>
<tr>
<td>-----------------------------------</td>
</tr>
<tr>
<td>IP header of</td>
</tr>
<tr>
<td>original L3 packet</td>
</tr>
<tr>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Upper layer headers and</td>
</tr>
<tr>
<td>leading portion of body</td>
</tr>
<tr>
<td>of the original L3 packet</td>
</tr>
<tr>
<td>-----------------------------------</td>
</tr>
</tbody>
</table>
```

Figure 4: AERO Interface L2 Error Message Format
The AERO node rules for processing these L2 error messages is as follows:

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

- 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 3.13.2).

- 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.

- 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.

- 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).

- 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.

- 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
Internet-Draft                    AERO                     February 2015

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:

```
+-------------------------------+
<p>|     L3 IP Header of           |</p>
<table>
<thead>
<tr>
<th>error message</th>
</tr>
</thead>
</table>
+-------------------------------+
<table>
<thead>
<tr>
<th>L3 ICMP Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP header of</td>
</tr>
<tr>
<td>original L3 packet</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
<tr>
<td>Upper layer headers and</td>
</tr>
<tr>
<td>leading portion of body</td>
</tr>
<tr>
<td>of the original L3 packet</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
</tbody>
</table>
```

Figure 5: 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 a service such as the Lightweight Directory Access Protocol (LDAP) [RFC4511] or a similar prefix/host reservation system.

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 a Vendor-Specific Information Option 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 DHCPv6 PD request [RFC3315][RFC3633] with 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 Client also includes 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 includes an AERO Link-Layer Address Request (ALLAREQ) option with the format shown in Figure 6:

```
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          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   enterprise-number (=45282)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 0   |    Reserved   |    Link ID    |   Preference  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: AERO Link-Layer Address Request (ALLAREQ) Option
In the above format, the Client sets ‘option-code’ to OPTION_VENDOR_OPTS, sets ‘option-len’ to 8, sets ‘enterprise-number’ to 45282 (see: IANA Considerations), sets ‘Type’ to 0 to indicate "AERO Link-Layer Address Request (ALLAREQ) option, and sets ‘Reserved’ to 0. The Client then sets appropriate ‘Link ID’ and ‘Preference’ values for the underlying interface over which the DHCPv6 PD request will be issued the same as for an S/TLLAO as shown in Figure 2. (The Client MAY instead omit the ALLAREQ option; in that case, the Server considers the message the same as if the Client had inserted an ALLAREQ option with ‘Link ID’ set to 0 and ‘Preference’ set to 255.) The Client finally includes any necessary authentication options to identify itself to the DHCPv6 server, and sends the encapsulated DHCPv6 PD request via the underlying interface.

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. If the Reply message contains an AERO Service Prefix Advertisement (ASPADV) option (see: Section 3.15.3) the Client also caches each ASP in the option. The Client then considers the link-layer address of the Server as the primary default encapsulation address for forwarding packets for which there is no more-specific forwarding information. The Client can also examine the UDP Port Number and IP Address in the AERO Link-Layer Address Reply (ALLAREP) option of the Reply message to determine it’s link-layer address from the perspective of the Server, which may be different than from its own perspective (see: Section 3.15.3).

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

On some platforms (e.g., popular cell phone operating systems), the act of assigning a default IPv6 route 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:

- the IPv6 source address is the Client’s AERO address
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].

The Client subsequently renews its ACP delegation through each of its Servers by performing DHCPv6 Renew/Reply exchanges with its AERO address as the IPv6 source address, ‘All_DHCP_Relay_Agents_and_Servers’ as the IPv6 destination address, the link-layer address of a Server as the link-layer destination address and the same Client identifier, authentication options and ALLAREQ as was 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. If the Client sends synthesized RA and/or DHCPv4 messages (see above), it also sends a new synthesized message when issuing a DHCPv6 Renew or when re-initiating 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).

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 message on its AERO interface, it first 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.

The Server then delegates the ACP and creates an IP forwarding table entry so that the AERO routing system will propagate the ACP to all Relays (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 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.

The Server also includes an AERO Link-Layer Address Reply (ALLAREP) option filled out with the UDP Port Number and IP Address values it observed when it received the ALLAREQ in the Client’s original DHCPv6 message. The ALLAREP option is formatted as shown in Figure 7:

```
 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 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| enterprise-number (=45282) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Type = 1 | Reserved | Link ID | Preference |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| UDP Port Number |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| IP Address |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: AERO Link-Layer Address Reply (ALLAREP) Option

In the ALLAREP, the Server sets ‘option-code’ to OPTION_VENDOR_OPTS, sets ‘option-length’ to 26, sets ‘enterprise-number’ to 45282, sets ‘Type’ to 1 and sets ‘Reserved’ to 0. Next, the Server sets ‘Link
ID’ and ‘Preference’ to the same values that appeared in the ALLAREQ, and sets ‘UDP Port Number’ and ‘IP address’ to the Client’s link-layer address. Note that if the Client did not include an ALAREQ option in its DHCPv6 message, the Server MUST still include an ALLAREP option in the corresponding reply with ‘Link ID’ set to 0 and ‘Preference’ set to 255.

The Server next includes an AERO Service Prefix Advertisement Option (ASPADV) formatted as shown in Figure 8:

```
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 |
+-------------------------------------------+
| enterprise-number (=45282) |
+-------------------------------------------+
| Type = 2 | Reserved | Prefix Length |
+-------------------------------------------------------------------+
| ASP (1) |
+-------------------------------------------------------------------+
| Reserved | Prefix Length |
+-------------------------------------------------------------------+
| ASP (2) |
+-------------------------------------------------------------------+
| Reserved | Prefix Length |
+-------------------------------------------------------------------+
| ASP (3) |
+-------------------------------------------------------------------+
| (etc.) |
+-------------------------------------------------------------------+
```

Figure 8: AERO Service Prefix Advertisement (ASPADV) Option

In the ASPADV, the Server sets ‘option-code’ to OPTION_VENDOR_OPTS, sets ‘option-length’ to the length of the option, sets ‘enterprise-number’ to 45282, sets ‘Type’ to 2 and sets ‘Reserved’ to 0. Next, the Server 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). For IPv6, valid values for the Prefix Length field are 0 through 64; for IPv4, valid values are 0 through 32.
When the Server’s DHCPv6 function admits the DHCPv6 Reply message into the AERO interface, the interface 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 ALLAREP option. The AERO interface then uses the Client link-layer address information in the ALLAREP option as the link-layer address for encapsulation.

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 from the perspective of DHCPv6. However, in some implementations the DHCPv6 server and AERO interface driver may be located in separate modules that preclude information sharing through standard APIs. In that case, the AERO interface driver module can act as a Lightweight DHCPv6 Relay Agent (LDRA) [RFC6221] "bump in the wire" to include an ALLAREP option and any other options in a Relay-Forward message encapsulation to the DHCPv6 Server. The AERO interface driver prepares an ALLAREP option with the 'UDP Port Number' and 'IP Address' taken from the Client’s link-layer address and writes the values found in the ALLAREQ option in the 'Link ID' and 'Preference' fields. The AERO interface driver then wraps the ALLAREP option in a Relay-Supplied DHCP Option [RFC6422], 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 ALLAREP option and authenticates the encapsulated DHCPv6 message. When the DHCPv6 server prepares a Reply message, it then includes the ALLAREP 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 AERO interface driver, where the Relay-Reply wrapper is discarded through the application of LDRA and the DHCPv6 message is delivered to the Client.
3.16. AERO Forwarding Agent Discovery

AERO Relays, Servers and Clients MAY associate with one or more companion AERO Forwarding Agents as platforms for offloading high-speed data plane traffic. AERO nodes distribute forwarding information to Forwarding Agents via an out-of-band messaging service (e.g., NETCONF [RFC6241], etc.).

When an AERO node receives a data packet on an AERO interface with a network layer destination address for which it has distributed forwarding information to one or more Forwarding Agents, the node returns an RA message to the source neighbor (subject to rate limiting) then forwards the data packet as usual. The RA message includes one or more SLLAOs with the link-layer addresses of candidate Forwarding Engines.

If the forwarding information pertains only to a specific ACP, the AERO node sets the network-layer source address of the RA to the AERO address corresponding to the ACP, and sets the default router lifetime to 0. If the forwarding information pertains to all addresses, the AERO node instead sets the network-layer source address of the RA to its own link-local address and sets the default router lifetime to a non-zero value.

When the source neighbor receives the RA message, it SHOULD record the link-layer addresses in the SLLAOs as the encapsulation addresses to use for sending subsequent data packets with addresses that match the information in the RA. However, the source MUST continue to use the primary link-layer address of the AERO node 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. 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 9 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)            |
|fe80::2001:db8:0:0|

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AERO Client C1</td>
</tr>
<tr>
<td>fe80::2001:db8:0:0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AERO Client C2</td>
</tr>
<tr>
<td>fe80::2001:db8:1:0</td>
</tr>
</tbody>
</table>

2001:db8:0::/48  2001:db8:1::/48
```

Figure 9: AERO Reference Operational Scenario

In Figure 9, 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 9, 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 processes 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/Predicted messages are formatted as shown
in Figure 10:

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
+-------------------------------+
| Type (=137) | Code (=0/1) | Checksum |
+-------------------------------+
| Reserved | Prefix Length | |
+-----------------------------+
|                             | Target Address |
|                             |                |
|                             |                |
+-----------------------------+
+-------------------------------+
|                             | Destination Address |
|                             |                     |
|                             |                     |
+-----------------------------+
|                             | Options ... |
+-----------------------------+

Figure 10: AERO Redirect/Predicted Message Format

3.17.4. Sending Predicted

When a Client forwards a packet with a source address from one of its
ACFs 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 Predicted 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 Predicted message
forward toward Client (‘C2’), subject to rate limiting (see
Section 8.2 of [RFC4861]). Client (‘C1’) prepares the Predicted
message as follows:
the link-layer source address is set to ‘L2(C1)’ (i.e., the link-layer address of Client (‘C1’)).

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

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

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

the Type is set to 137.

the Code is set to 1 to indicate "Predirect".

the Prefix Length is set to the length of the prefix to be assigned to the Target Address.

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

the Destination Address is set to the source address of the originating packet that triggered the Predirection event. (If the originating packet is an IPv4 packet, the address is constructed in IPv4-compatible IPv6 address format).

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

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

the message includes a Redirected Header Option (RHO) that contains the originating packet truncated 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 its 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:

- the link-layer source address is set to ‘L2(C2)’ (i.e., the link-layer address of Client (‘C2’)).
- the link-layer destination address is set to ‘L2(S2)’ (i.e., the link-layer address of Server (‘S2’)).
- the network-layer source address is set to fe80::2001:db8:1:0 (i.e., the AERO address of Client (‘C2’)).
- the network-layer destination address is set to fe80::2001:db8:0:0 (i.e., the AERO address of Client (‘C1’)).
- the Type is set to 137.
- the Code is set to 0 to indicate "Redirect".
- the Prefix Length is set to the length of the prefix to be applied to the Target Address.
- the Target Address is set to fe80::2001:db8:1:0 (i.e., the AERO address of Client (‘C2’)).
- the Destination Address is set to the destination address of the originating packet that triggered the Redirection event. (If the originating packet is an IPv4 packet, the address is constructed in IPv4-compatible IPv6 address format).
- the message includes one or more TLLAOs with Link ID and Preference set to appropriate values for Client (‘C2’)’s underlying interfaces, and with UDP Port Number and IP Address set to ‘0’.
The message SHOULD include a Timestamp option and MUST echo the Nonce option received in the Predirect (i.e., if a Nonce option is included).

The message includes as much of the RHO copied from the corresponding AERO Predirect message as possible such that at least the network-layer header is included but the size of the message does not exceed 1280 bytes.

After Client ('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 its 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 and with a CLLAO that includes the correct Link ID and Preference values. If authentication succeeds, the Server then updates 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 9, when Client (‘C2’) needs to change its link-layer address it sends unicast unsolicited NA messages to Client (‘C1’) via Server (‘S2’) as follows:

- The link-layer source address is set to ‘L2(C2)’ (i.e., the link-layer address of Client (‘C2’)).
- The link-layer destination address is set to ‘L2(S2)’ (i.e., the link-layer address of Server (‘S2’)).
- The network-layer source address is set to fe80::2001:db8:1:0 (i.e., the AERO address of Client (‘C2’)).
- 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 Preference 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 a CLLAO that includes the new Link ID and Preference values. If authentication succeeds, the Server then updates its neighbor cache and sends a DHCPv6 Reply.
The Client MAY then send unsolicited NA messages to each of its correspondent Clients to inform them of the new link-layer address as described in Section 3.19.1.

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 a CLLAO that includes the deprecated Link ID and a Preference value of 0. If authentication succeeds, the Server then updates its neighbor cache and sends a DHCPv6 Reply. The Client SHOULD then send unsolicited NA messages to each of its correspondent Clients to inform them of the deprecated link-layer address as described in Section 3.19.1.

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 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 Request message directly to the new Client and queues any arriving data packets addressed to the departed MN. The Request message includes the MN’s ID as the DUID, the ACP in an IA_PD option, the AERO address derived from the MN’s ACP as the network-layer source address, ‘All_DHCP_Relay_Agents_and_Servers’ as the network-layer destination address, 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 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 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 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 AERO address derived from the MN’s ACP as the network-layer source address, ‘All_DHCP_Relay_Agents_and_Servers’ as the network-layer destination address, 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 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.


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:

- The encapsulating IPv6 header source address is set to 2001:db8:1:: (i.e., the IPv6 subnet router anycast address for (‘C1’)s ACP)
- The encapsulating IPv6 header destination address is set to 2001:db8:2:: (i.e., the IPv6 subnet router anycast address for (‘C2’)s ACP)
- The encapsulating IPv6 header is followed by a UDP header with source and destination port set to 8060
- The encapsulated IPv6 header source address is set to fe80::2001:db8:1:0 (i.e., the AERO address for (‘C1’))
- The encapsulated IPv6 header destination address is set to fe80::2001:db8:2:0 (i.e., the AERO address for (‘C2’))
- 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:

- The encapsulating IPv6 header source address is set to 2001:db8:1000:1 (i.e., the Internet IPv6 address of (‘C1’))
The encapsulating IPv6 header destination address is set to 2001:db8:2:: (i.e., the IPv6 subnet router anycast address for ('C2')'s ACP).

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

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

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

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

An application-layer implementation is in progress.

5.  IANA Considerations

IANA has assign a 4-octet Private Enterprise Number "45282" for AERO in the "enterprise-numbers" registry. No further IANA actions are required.

6.  Security Considerations

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

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.

7. Acknowledgements

Discussions both on IETF lists and in private exchanges helped shape
some of the concepts in this work. Individuals who contributed
insights include Mikael Abrahamsson, 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, 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 Keith Bartley, 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, Yuelli 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:

- The Internet Routing Overlay Network (IRON)
  [ RFC6179 ][ I-D.templin-ironbis ]
Virtual Enterprise Traversal (VET)  
[RFC5558][I-D.templin-intarea-vet]

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

AERO, First Edition [RFC6706]

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

8. References

8.1. Normative References


8.2. Informative References


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

Email: fltemplin@acm.org
Abstract

In the basic Proxy Mobile IPv6 (PMIPv6) specification, a Mobile Node (MN) is assigned a 64-bit Home Network Prefix (HNP) during the initial attachment for the Home Address (HoA) configuration. During the movement of the MN, this prefix is unchanged and unnecessary for the MN to reconfigure the HoA. However, the current protocol does not specify related operations to support the MN to timely receive and configure a new HNP when the allocated HNP changes. In this draft, this problem is discussed and a possible solution is proposed based on RFC 7077.

Requirements Language

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].

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html
1. Introduction

Network managers currently prefer to Provider Independent (PI) addressing for IPv6 to attempt to minimize the need for future renumbering. However, widespread use of PI may create very serious BGP scaling problems. It is thus desirable to develop tools and practices that may make renumbering a simpler process to reduce demand for IPv6 PI space [1]. In this draft, we aims to solve the HNP renumbering problem when the HNP in PMIPv6 [2] is not a PI type.

Then the HNP renumbering may happen at least in the following three cases:

In the first case, the PMIPv6 service provider is assigned the HNP set from the (uplink) ISP, and then the HNP renumbering will happen if the PMIPv6 service provider switches to a different ISP.
In the second case, multiple Local Mobility Anchors (LMAs) may be deployed by the same PMIPv6 service provider, and then each LMA may serve for a specific HNP set. In this case, the HNP of a MN may change if the current serving LMA switches to another LMA but without inheriting the assigned HNP [3].

In the last case, the PMIPv6 HNP renumbering may be caused by the re-building of the network architecture as the companies split, merge, grow, relocate or reorganize. For example, the PMIPv6 service provider may reorganize its network topology.

For the Mobile IPv6 (MIPv6), when the home network prefix changes (maybe due to the above reasons), the Home Agent (HA) will actively notify the new prefix to the MN and then the renumbering of the HoA can be well supported [4]. While in the basic PMIPv6, the PMIPv6 binding is triggered by the Mobile Access Gateway (MAG), which detects the attachment of the MN. When the HNP renumbering happens in the first case or the LMA and HNP both changes in the second or third cases, a scheme is needed for the LMA to immediately initiate the PMIPv6 binding state refreshment. Although this issue is also discussed in the RFC 5213 (section 6.12), the related solution is not specified.

2. HNP renumbering support

RFC 7077 [5] specifies a scheme to support the asynchronously update from the LMA to the MAG about changes related to a mobility session. With this protocol, the HNP renumbering can be easily supported and the basic operation is summarized as following:

1) When the PMIPv6 service provider renumbers the HNP set or the serving LMA switches to another one but does not inherit the related HNP, the current LMA (or new LMA) will initiate the HNP renumbering operation. Firstly, it should allocate a new HNP for the related MN.

2) The LMA sends the Update Notification (UPN) message to the MAG. In the UPN message, the Notification reason is set to 2 (UPDATE-SESSION-PARAMETERS). Besides, HNP option containing the new HNP and the Mobile Node Identifier option carrying MN'ID are contained as Mobility Options of UPN.

3) After the MAG receives this UPN message, it will recognize that the related MN has a new HNP now. Then the MAG sends the old HNP in the RA with zero-valued lifetime to the MN and sends the new HNP in the RA with lifetime larger than zero.
4) Besides, the MAG sends back the Update Notification Acknowledgement (UNA) to the LMA for the notification of successful update of the related binding state, routing state and RA settings.

5) For the MN, it deletes the old HoA due to the zero-valued lifetime RA advertisement and configures a new HoA with the meaningful HNP.

The detailed protocol operation and signaling message extensions will be specified further.

3. DNS update

In order to maintain the reachability of the MN, the DNS resource record corresponding to this MN may need to be updated when the HNP of MN changes. Although this operation in PMIPv6 has not been specified by the current protocols, we here list two important issues to be considered for this operation.

1) Since the DNS update must be performed securely in order to prevent attacks or modifications from malicious ones, the node performing this update must share a security association with the DNS server [6]. If the MN does not share a security association with the DNS server and the DNS entry update can be performed by the network entities, such as Authentication, Authorization and Accounting (AAA) server or LMA.

2) For the dynamic update, another important issue should be considered is the TTL setting when the HNP renumbering is possible. The TTL should be set according to the possible lifetime of the HNP.

4. Security Considerations

The security considerations in PMIPv6 protocol are enough for the basic operation of this draft.

Besides, the security dynamic DNS should be supported whatever the DNS update is executed by network entities or MN itself. In other words, the security association should always be established between the DNS server and updater.

Other security issues will be added further.

5. References


Authors’ Addresses

Zhiwei Yan
China Internet Network Information Center
No.4 South 4th Street, Zhongguancun
Beijing, P. R. China
Email: yan@cnnic.cn

Jong-Hyouk Lee
Department of Computer Science and Engineering
Sangmyung University
31, Sangmyeongdae-gil, Dongnam-gu
Cheonan, Republic of Korea
E-mail: jonghyouk@smu.ac.kr

Xiaodong Lee
China Internet Network Information Center
No.4 South 4th Street, Zhongguancun
Beijing, P. R. China
Email: xl@cnnic.cn

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.