Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 11, 2015

S. Matsushima
Softbank Telecom
R. Wakikawa
Softbank Mobile

March 10, 2015

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

Copyright Notice

Table of Contents

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.

```
                                   _+---+_ _ _+---+_ _ _
EPC                              / | S |       | P |     /
Control-plane                   /  | G |       | G |    /
                               /_ _| W |_ _ _ | W |__/
            +---+   +---+          |   |       |   |
            |   |   | e |          |   |       |   |
            | U |   | N |        _|   |_ _ _ |   |_ _ _
EPC         | E |   | B |       / |   |       |   |     /
User-plane  +---+   +---+      /  +---+     +---+    /
                               /_ _ _Existing EPC_ _/


                                   _+---+_ _ _+---+_ _ _
vEPC                             / | S |       | P |     /
Control-plane                   /  | G |       | G |    /
                               /_ _| W |_ _ _ | W |__/
                                     |   |       |   |
            +---+   +---+          +---+     +---+
            |   |   | e |            +----------+
            | U |   | N |          _|IP Routing|_ _ _
Stateless   | E |   | B |         /  | Network  |     /
User-plane  +---+   +---+        /   +----------+    /
                                /_ _ _ _ _ _ _ _ _ /
```

NFV enabled EPC architecture

Figure 1

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 hyper-visor 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.

Flat Design for Distributed Operation

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 tunneled to SGW. SGW then forwards the packet to UE via access points (i.e. eNodeB) over Radio Area Network (RAN).

```
                                  .--.
                                _(    )_
                              _(        )
            +---+    +---+  (   EPC-NW    )  +---+
    [UE]~~~~|RAN|====|SGW|==================|PGW| -> the Internet
            +---+    +---+  `--(______)---'  +---+
                       <--------Tunneling------>

    ~~~ Wireless Link
    === GTP Tunnel (PMIP tunnel)
    --- IP link
```

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.

```
            [EPC-E anycast address]
                  |             .--.
                  |           _(    )_
                  |  +---+   _(        )
            +---+  \ |EPC|  (  IPv6 Core  )  +---+
    [UE]~~~~|RAN|====| -E|-( `    NW      )-|RTR| --> the Internet
            +---+    +---+  ( `         ) )  +---+
          (eNodeB)            `--(_____)--'
                       <--------Routing-------->
```

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 utilizes 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 states into the BGP speaker.

BGP speaker in the vEPC should utilize FPCP Client function to receive exported information from the the agent. When the BGP speaker gets those information, they extract forwarding policies from them and generate BGP UPDATE messages that contains 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.

```
                        vEPC(MME/SGW
      UE        eNodeB   PGW/HSS/AAA)   EPC-E        RTR
      |           |           |           |           |
      |---------->|(1)        |           |           |
                  |---------->|(1) MME    |           |
      |/--------------------\|           |           |
      |  Authentication     |(2) AAA    |           |
      |\--------------------/|           |           |
      |           |           |--+SGW/PGW |           |
      |           |           |  |(3)(4)  |(4)SGW's TEID is configured
      |           |           |<-+        |   and notified to MME
      |           |<----------|(5) MME    |           |
      |/---------\|           |           |           |
      | RB setup  |(6)        |           |           |
      |\---------/|           |           |           |
      |           |---------->|(7) MME    |           |
      |---------->|(8)        |           |           |
      |           |---------->|(9) MME    |(9)eNodeB TEID is configured
      |           |           |           |   and sent to MME
      |= = = = = = = Uplink Data = = = = =>= = = = ==>|(10)
      |       The uplink is not yet built here        |
      |           |           |           |           |
      |           |           |--+SGW/MME |           |
      |           |           |  |(11,12) |           |
      |           |           |<-+        |           |
      |<= = = = = = Downlink Data = = = = <= = = = = =|(13)
      |       The downlink is not yet built here      |
      |           |           |           |           |
      |           |           |---------->|(14) Route Update
      |           |           |  [Dst: UE-prefix,
      |           |           |   NxtHop: S1-U addr and TEID of enodeB
      |           |           |           |           |
      |           |           |           |---------->|(15)Route update
      |           |           |           |[Dst: UE-prefix,
      |           |           |           | NxtHop: EPC-E address]
      |           |           |           |           |
      |---------RS or DHCP Request------->|(16)       |
      |<--------RA or DHCP Reply----------|(17)       |
```

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.

```
|          n bits           | o bits (=< 32)|64-n-o bits|
+---------------------------+---------------+-----------+
|        PDN Prefix         |      TEID     | subnet ID |
+---------------------------+---------------+-----------+
<---------------------------64bits---------------------->
```

Stateless-pd Prefix

Figure 5

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.

```
                Source      Target
        UE       eNodeB      eNodeB        vEPC      EPC-E     RTR
        |          |          |              |         |        |
        |          |          |              |         |        |
        |          |          |              |         |        |
        |          ---Handover Required---->|(1)      |        |
        |          |          |<--Handover---|(2)      |        |
        |          |          |   Request    |         |        |
        |          |          |              |         |        |
        |          |          |--Handover--->|(3)      |        |
        |          |          |  Acknoledge  |         |        |
        |          |          |              |         |        |
        |<---------|<----Handover Command----|(4)      |        |
        --Handover Confirm-->|(5)            |         |        |
        |= = = = = = = = = Uplink Data = = = = = = = = =>= = ==> |(6)
        |          |          |              |         |        |
        |          |          |---Handover-->|(7)      |        |
        |          |          |   Notify     |         |        |
        |          |          |              |-------->|(8)     |
        |          |          |              | Route Update      |
        |          |          |              |[Dst   : UE-prefix,
        |          |          |              | NxtHop: Target eNode's
        |          |          |              |         S1-U addr/TEID]
        |          |          |              |         |        |
        |<== = = = = = = = = Downlink Data = = = = = =  =<== = = =|(9)
        |          |          |              |         |        |
```

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.

```
            +--------------------+
            | Correspondent Node |
            +--------------------+
                      |
                   .--.        [Route on the Internet]
                 _(.   `)      Destination: Operator's Prefix
               _(        `)_   NextHop: RTR-1/2
              (  Internet   `)
             ( `  .        )  )
              `--(________)---'
                /         \     [Route at RTR]
             +----+    +----+  Destination: PDN Prefix
             |RTR1|    |RTR2|  NextHop: EPC-E2-b
             +----+    +----+
                 \ .--. /
                 _( IP `.
                ( CORE NW )
              ___( `  .  )  )__
             /   `--(___.--'   \
            /         |         \
            |a         |b         |c   [Route at EPC-E]
         +---+--+   +---+--+   +---+--+ Destination: UE's Prefix *1
         |EPC-E1|   |EPC-E2|   |EPC-E3| NextHop: GTP tunnel (eNB-C) *2
         +---+--+   +---+--+   +---+--+
            |X         |X         |X
             \       .--.        /
              \_____( RAN `.____/
                (  ACCESS )
             ____( `  NW  )  )________
           /     `--(___.--'          \
          /        |         |         \
         |1        |2        |3        |4
       +--+--+   +--+--+   +--+--+   +--+--+
       |eNB-A|   |eNB-B|   |eNB-C|   |eNB-D|
       +--+--+   +--+--+   +--+--+   +--+--+
          :         :         :         :
                                UE  -->   UE'
   *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.
```

Routing Mechanism Overview

Figure 7

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.

Bearer type	UE function	EPC-E function	Gateway function
IPv4	-	B4	AFTR
IPv4	-	CLAT	PLAT
IPv6	MAP-CE	-	MAP-BR
IPv6	B4	-	AFTR
IPv6	CLAT	-	PLAT

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.

```
                                  .--.
                        EPC-E      _(    `)_     Gateway
                        +----+   _(          `)   +------+
                        | B4 |  (  IPv6 Core  )   | AFTR |
          [UE]==IPv4-only==| or |-( `    NW       )-| or   | --> Internet
                  bearer  |CLAT|  ( `          ) )  | PLAT |     (IPv4)
                        +----+   `--(_____)--'   +------+
                          <-----IPv4 over IPv6----->
                                      or
                              v4v6 translation
```

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 ECP-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.

[RFC5512] Mohapatra, P. and E. Rosen, "The BGP Encapsulation 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.

[RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, January 2005.

[RFC5213] Gundavelli, S., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, August 2008.

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

[RFC5844] Wakikawa, R. and S. Gundavelli, "IPv4 Support for Proxy Mobile IPv6", RFC 5844, May 2010.

[RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

[RFC6301] Zhu, Z., Wakikawa, R., and L. Zhang, "A Survey of Mobility 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.

[RFC6459] Korhonen, J., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, January 2012.

[RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", RFC 6877, April 2013.

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

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2015

D. Moses
Intel
A. Yegin
Samsung

March 8, 2015

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

Copyright Notice

Table of Contents

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 trough out 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

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

[RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, September 2007.

[RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

6.2. Informative References

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

Authors' Addresses

Danny Moses
Intel
Petah Tikva
Israel

Email: danny.moses@intel.com

Alper Yegin
Samsung
Istanbul
Turkey

Email: alper.yegin@partner.samsung.com

DMM WG
Internet Draft
Intended status: Standard Track
Expires: August 26, 2015

Seil Jeon
Institute de Telecomunicacoes
S. Figueiredo
Universidade de Aveiro
Younghan Kim
Soongsil University

February 26, 2015

Use Cases and API Extension for Source IP address selection
draft-sijeon-dmm-use-cases-api-source-00.txt

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

Copyright Notice

Table of Contents

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 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

[RFC5014] E. Nordmark, S. Chakrabarti, and J. Laganier, "IPv6 Socket API for Source Address Selection," IETF RFC 5014, Sep. 2007.

[RFC6724] D. Thaler, R. Draves, A. matsumoto, and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)," IETF RFC 6724, Sep. 2012.

[draft-yegin-dmm-ondemand-mobility] A. Yegin, K. Kweon, J. Lee, and J. Park, "On Demand Mobility Management," draft-yegin-dmm-ondemand-mobility-02, Jul. 2014.

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

Network Working Group
Internet-Draft
Obsoletes: rfc5320, rfc5558, rfc5720,
rfc6179, rfc6706 (if
approved)
Intended status: Standards Track
Expires: November 11, 2018

F. Templin, Ed.
Boeing Research & Technology
May 10, 2018

Asymmetric Extended Route Optimization (AERO)
draft-templin-aerolink-82.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 route optimization services for improved performance. AERO provides an IPv6 link-local address format that supports operation of the IPv6 Neighbor Discovery (ND) protocol and links ND to IP forwarding. Dynamic link selection, mobility management, quality of service (QoS) signaling and route optimization are naturally supported through dynamic neighbor cache updates, while IPv6 Prefix Delegation (PD) is supported by network services such as the Dynamic Host Configuration Protocol for IPv6 (DHCPv6). AERO is a widely-applicable tunneling solution especially well-suited to aviation services, mobile Virtual Private Networks (VPNs) and other applications as described in this document.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document specifies the operation of IP over tunnel virtual links using Asymmetric Extended Route Optimization (AERO). The AERO link can be used for tunneling between 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 route optimization services for improved performance [RFC5522].

AERO provides an IPv6 link-local address format that supports operation of the IPv6 Neighbor Discovery (ND) [RFC4861] protocol and links ND to IP forwarding. Dynamic link selection, mobility management, quality of service (QoS) signaling and route optimization are naturally supported through dynamic neighbor cache updates, while IPv6 Prefix Delegation (PD) is supported by network services such as the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC3315][RFC3633].

A node's AERO interface can be configured over multiple underlying interfaces. From the standpoint of ND, AERO interface neighbors therefore may appear to have multiple link-layer addresses (i.e., the addresses assigned to underlying interfaces). Each link-layer address is subject to change due to mobility and/or QoS fluctuations, and link-layer address changes are signaled by ND messaging the same as for any IPv6 link.

AERO is applicable to a wide variety of use cases. For example, it can be used to coordinate the Virtual Private Network (VPN) links of mobile nodes (e.g., cellphones, tablets, laptop computers, etc.) that connect into a home enterprise network via public access networks using services such as OpenVPN [OVPN]. AERO is also applicable to aviation services for both manned and unmanned aircraft where the aircraft is treated as a mobile node that can connect an Internet of Things (IoT). Other applicable use cases are also in scope.

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:

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

IPv6 Prefix Delegation (PD)
: a networking service for delegating IPv6 prefixes to nodes on the link. The nominal PD service is DHCPv6 [RFC3315] [RFC3633], however other services (e.g., alternate ND options, network management, static configuration, etc.) are also possible.

(native) Internetwork
a connected IPv6 or IPv4 network topology over which the AERO link virtual overlay is configured and native peer-to-peer communications are supported. Example Internetworks include the global public Internet, private enterprise networks, aviation networks, etc.

AERO link
a Non-Broadcast, Multiple Access (NBMA) tunnel virtual overlay configured over an underlying Internetwork. All nodes on the AERO link appear as single-hop neighbors from the perspective of the virtual overlay even though they may be separated by many underlying Internetwork hops. The AERO mechanisms can also operate over native link types (e.g., Ethernet, WiFi etc.) when a tunnel virtual overlay is not needed.

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

AERO address
an IPv6 link-local address constructed as specified in Section 3.4.

AERO node
a node that is connected to an AERO link.

AERO Client ("Client")
a node that requests IP PDs from one or more AERO Servers. Following PD, the Client assigns an AERO address to the AERO interface for use in ND exchanges with other AERO nodes. A node that acts as an AERO Client on one AERO interface can also act as an AERO Server on a different AERO interface.

AERO Server ("Server")
a node that configures an AERO interface to provide default forwarding services for AERO Clients. The Server assigns an administratively-provisioned IPv6 link-local address to the AERO interface to support the operation of the ND/PD services. An AERO Server can also act as an AERO Relay.

AERO Relay ("Relay")
a node that configures an AERO interface to relay IP packets between nodes on the same AERO link and/or forward IP packets between the AERO link and the native Internetwork. The Relay

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

AERO Proxy ("Proxy")
a node that provides proxying services for Clients that cannot associate directly with Servers, e.g., when the Client is located in a secured internal enclave and the Server is located in the external Internetwork. The AERO Proxy is a conduit between the secured enclave and the external Internetwork in the same manner as for common web proxies, and behaves in a similar fashion as for ND proxies [RFC4389].

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

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

underlying network
the same as defined for Internetwork.

underlying link
a link that connects an AERO node to the underlying network.

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

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. Packets transmitted over an AERO interface use link-layer addresses as encapsulation header source and destination addresses. Destination link-layer addresses can be either "reachable" or "unreachable" based on dynamically-changing network conditions.

network layer address
the source or destination address of an 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. The Client sees each EUN as a "downstream" network and sees the

AERO interface as its point of attachment to the "upstream" network.

AERO Service Prefix (ASP)
an IP prefix associated with the AERO link and from which more-specific AERO Client Prefixes (ACPs) are derived.

AERO Client Prefix (ACP)
an IP prefix derived from an ASP and delegated to a Client, where the ACP prefix length must be no shorter than the ASP prefix length and must be no longer than 64 for IPv6 or 32 for IPv4.

base AERO address
the lowest-numbered AERO address from the first ACP delegated to the Client (see Section 3.4).

Throughout the document, the simple terms "Client", "Server", "Relay" and "Proxy" refer to "AERO Client", "AERO Server", "AERO Relay" and "AERO Proxy", respectively. Capitalization is used to distinguish these terms from DHCPv6 client/server/relay [RFC3315].

The terminology of DHCPv6 [RFC3315][RFC3633] and IPv6 ND [RFC4861] (including the names of node variables, messages and protocol constants) is used throughout this document. Also, the term "IP" is used to generically refer to either Internet Protocol version, i.e., IPv4 [RFC0791] or IPv6 [RFC8200].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. Lower case uses of these words are not to be interpreted as carrying RFC2119 significance.

3. Asymmetric Extended Route Optimization (AERO)

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

3.1. AERO Link Reference Model

```
                              .-(::::::::)
                            .-(::::::::::::)-.
                           (:: Internetwork ::)
                            `-(::::::::::::)-'
                               `-(::::::)-'
                                   |
       +--------------+   +--------+-------+   +--------------+
       |AERO Server S1|   | AERO Relay R1  |   |AERO Server S2|
       |  Nbr: C1, R1 |   |   Nbr: S1, S2  |   |  Nbr: C2, R1 |
       |  default->R1 |   |(X1->S1; X2->S2)|   |  default->R1 |
       |    X1->C1    |   |     ASP A1     |   |    X2->C2    |
       +-------+------+   +--------+-------+   +------+-------+
               |    AERO Link      |                  |
       X---+---+-------------------+-+----------------+---+---X
           |                         |                  |
     +-----+--------+     +----------+------+    +--------+-----+
     |AERO Client C1|     |  AERO Proxy P1  |    |AERO Client C2|
     |    Nbr: S1   |     |(Proxy Nbr Cache)|    |   Nbr: S2    |
     | default->S1  |     +--------+--------+    | default->S2  |
     |    ACP X1    |              |              |    ACP X2    |
     +------+-------+     .--------+------.      +-----+--------+
            |           (- Proxyed Clients -)          |
          .-.             `---------------'          .-.
        ,-(  _)-.                                  ,-(  _)-.
     .-(_  IP   )-.   +-------+     +-------+    .-(_  IP   )-.
    (__    EUN      )--|Host H1|     |Host H2|--(__    EUN      )
      `-(______)-'    +-------+     +-------+      `-(______)-'
```

Figure 1: AERO Link Reference Model

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

- o AERO Relay R1 aggregates AERO Service Prefix (ASP) A1, acts as a default router for its associated Servers (S1 and S2), and connects the AERO link to the rest of the Internetwork.

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

- o AERO Clients C1 and C2 associate with Servers S1 and S2, respectively. They receive AERO Client Prefix (ACP) delegations X1 and X2, and also act as default routers for their associated physical or internal virtual EUNs. Simple hosts H1 and H2 attach to the EUNs served by Clients C1 and C2, respectively.

- o AERO Proxy P1 provides proxy services for AERO Clients in secured enclaves that cannot associate directly with other AERO link neighbors.

Each node on the AERO link maintains an AERO interface neighbor cache and an IP forwarding table the same as for any link. Although the figure shows a limited deployment, in common operational practice there may be many additional Relays, Servers, Clients and Proxies.

3.2. AERO Node Types

AERO Relays provide default forwarding services to AERO Servers. Each Relay also peers with Servers and other Relays in a dynamic routing protocol instance to discover the list of active ACPs (see Section 3.3). Relays forward packets between neighbors connected to the same AERO link and also forward packets between the AERO link and the native Internetwork. Relays present the AERO link to the native Internetwork as a set of one or more AERO Service Prefixes (ASPs) and serve as a gateway between the AERO link and the Internetwork. Relays maintain AERO interface neighbor cache entries for Servers, and maintain an IP forwarding table entry for each AERO Client Prefix (ACP). AERO Relays can also be configured to act as AERO Servers.

AERO Servers provide default forwarding services to AERO Clients. Each Server also peers with Relays in a dynamic routing protocol instance to advertise its list of associated ACPs (see Section 3.3). Servers facilitate PD exchanges with Clients, where each delegated prefix becomes an ACP taken from an ASP. Servers forward packets between AERO interface neighbors, and maintain AERO interface neighbor cache entries for Relays. They also maintain both neighbor cache entries and IP forwarding table entries for each of their associated Clients. AERO Servers can also be configured to act as AERO Relays.

AERO Clients act as requesting routers to receive ACPs through PD exchanges with AERO Servers over the AERO link. Each Client can 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. Clients maintain an AERO interface neighbor cache entry for each of their associated Servers as well as for each of their correspondent Clients.

AERO Proxies provide a conduit for AERO Clients connected to secured enclaves to associate with AERO link Servers. The Proxy can either be explicit or transparent. In the explicit case, the Client sends all of its control plane messages addressed to the Server to the link-layer address of the Proxy. In the transparent case, the Client sends all of its control plane messages to the Server’s link-layer address and the Proxy intercepts them before they leave the secured

enclave. In both cases, the Proxy forwards the Client's control and data plane messages to and from the Client's current Server(s). The Proxy may also discover a more direct route toward a target destination via AERO route optimization, in which case future outbound data packets would be forwarded via the more direct route. The Proxy function is specified in Section 4.

3.3. AERO Routing System

The AERO routing system comprises a private instance of the Border Gateway Protocol (BGP) [RFC4271] that is coordinated between Relays and Servers and does not interact with either the public Internet BGP routing system or the native Internetwork routing system. Relays advertise only a small and unchanging set of ASPs to the native Internetwork routing system instead of the full dynamically changing set of ACPs.

In a reference deployment, each AERO Server is configured as an Autonomous System Border Router (ASBR) for a stub Autonomous System (AS) using an AS Number (ASN) that is unique within the BGP instance, and each Server further uses eBGP to peer with one or more Relays but does not peer with other Servers. All Relays are members of the same hub AS using a common ASN, and use iBGP to maintain a consistent view of all active ACPs currently in service.

Each Server maintains a working set of associated ACPs, and dynamically announces new ACPs and withdraws departed ACPs in its eBGP updates to Relays. Clients are expected to remain associated with their current Servers for extended timeframes, however Servers SHOULD selectively suppress updates for impatient Clients that repeatedly associate and disassociate with them in order to dampen routing churn.

Each Relay configures a black-hole route for each of its ASPs. By black-holing the ASPs, the Relay will maintain forwarding table entries only for the ACPs that are currently active, and packets destined to all other ACPs will correctly incur Destination Unreachable messages due to the black hole route. Relays do not send eBGP updates for ACPs to Servers, but instead only originate a default route. In this way, Servers have only partial topology knowledge (i.e., they know only about the ACPs of their directly associated Clients) and they forward all other packets to Relays which have full topology knowledge.

Scaling properties of the AERO routing system are limited by the number of BGP routes that can be carried by Relays. At the time of this writing, the global public Internet BGP routing system manages more than 500K routes with linear growth and no signs of router

resource exhaustion [BGP]. Network emulation studies have also shown that a single Relay can accommodate at least 1M dynamically changing BGP routes even on a lightweight virtual machine, i.e., and without requiring high-end dedicated router hardware.

Therefore, assuming each Relay can carry 1M or more routes, this means that at least 1M Clients can be serviced by a single set of Relays. A means of increasing scaling would be to assign a different set of Relays for each set of ASPs. In that case, each Server still peers with one or more Relays, but the Server institutes route filters so that it only sends BGP updates to the specific set of Relays that aggregate the ASP. For example, if the ASP for the AERO link is 2001:db8::/32, a first set of Relays could service the ASP segment 2001:db8::/40, a second set of Relays could service 2001:db8:0100::/40, a third set could service 2001:db8:0200::/40, etc.

Assuming up to 1K sets of Relays, the AERO routing system can then accommodate 1B or more ACPs with no additional overhead for Servers and Relays (for example, it should be possible to service 1B /64 ACPs taken from a /34 ASP and even more for shorter prefixes). In this way, each set of Relays services a specific set of ASPs that they advertise to the native Internetwork routing system, and each Server configures ASP-specific routes that list the correct set of Relays as next hops. This arrangement also allows for natural incremental deployment, and can support small scale initial deployments followed by dynamic deployment of additional Clients, Servers and Relays without disturbing the already-deployed base.

Note that in an alternate routing arrangement each set of Relays could advertise an aggregated ASP for the link into the native Internetwork routing system even though each Relay services only smaller segments of the ASP. In that case, a Relay upon receiving a packet with a destination address covered by the ASP segment of another Relay can simply tunnel the packet to the other Relay. The tradeoff then is the penalty for Relay-to-Relay tunneling compared with reduced routing information in the native routing system.

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

3.4. AERO Interface Link-local Addresses

AERO interface link-local address types include administratively-provisioned addresses and AERO addresses.

Administratively-provisioned addresses are allocated from the range fe80::/96 and assigned to a Server or Relay's AERO interface.

Administratively-provisioned addresses MUST be managed for uniqueness by the administrative authority for the AERO link. The address fe80:: is reserved as the IPv6 link-local subnet router anycast address, and the address fe80::ffff:ffff is reserved as the "prefix-solicitation" address used by Clients to bootstrap AERO address autoconfiguration. These reserved addresses are therefore not available for general assignment.

An AERO address is an IPv6 link-local address with an embedded prefix based on an ACP and associated with a Client's AERO interface. AERO addresses remain stable as the Client moves between topological locations, i.e., even if its link-layer addresses change.

For IPv6, AERO addresses begin with the prefix fe80::/64 and include in the interface identifier (i.e., the lower 64 bits) a 64-bit prefix taken from one of the Client's IPv6 ACPs. For example, if the AERO Client receives the IPv6 ACP:

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

it constructs its corresponding AERO addresses as:

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

fe80::2001:db8:1000:2001

fe80::2001:db8:1000:2002

... etc. ...

fe80::2001:db8:1000:20ff
```

For IPv4, AERO addresses are based on an IPv4-mapped IPv6 address [RFC4291] formed from an IPv4 ACP and with a Prefix Length of 96 plus the ACP prefix length. For example, for the IPv4 ACP 192.0.2.32/28 the IPv4-mapped IPv6 ACP is:

```
0:0:0:0:0:FFFF:192.0.2.16/124
```

The Client then constructs its AERO addresses with the prefix fe80::/64 and with the lower 64 bits of the IPv4-mapped IPv6 address in the interface identifier as:

```
fe80::FFFF:192.0.2.16

fe80::FFFF:192.0.2.17

fe80::FFFF:192.0.2.18
```

```
... etc. ...

fe80:FFFF:192.0.2.31
```

When the Server delegates ACPs to the Client, both the Server and Client use the lowest-numbered AERO address from the first ACP delegation as the "base" AERO address (for example, for the ACP 2001:db8:1000:2000::/56 the base AERO address is fe80::2001:db8:1000:2000). The Client then assigns the base AERO address to the AERO interface and uses it for the purpose of maintaining the neighbor cache entry. The Server likewise uses the AERO address as its index into the neighbor cache for this Client.

If the Client has multiple AERO addresses (i.e., when there are multiple ACPs and/or ACPs with short prefix lengths), the Client originates ND messages using the base AERO address as the source address and accepts and responds to ND messages destined to any of its AERO addresses as equivalent to the base AERO address. In this way, the Client maintains a single neighbor cache entry that may be indexed by multiple AERO addresses.

3.5. AERO Interface Characteristics

AERO interfaces use encapsulation (see: Section 3.9) to exchange packets with neighbors attached to the AERO link.

AERO interfaces maintain a neighbor cache for tracking per-neighbor state the same as for any interface. AERO interfaces use ND messages including Neighbor Solicitation (NS), Neighbor Advertisement (NA), Router Solicitation (RS), Router Advertisement (RA) and Redirect for neighbor cache management. AERO interfaces use RS/RA messages with an embedded PD message (e.g., see: [I-D.templin-6man-dhcpv6-ndopt]). AERO interfaces include routing information in ND messages to support route optimization.

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |   Length = 5  |X|         Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Interface ID         |        UDP Port Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                           IP Address                          +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P00|P01|P02|P03|P04|P05|P06|P07|P08|P09|P10|P11|P12|P13|P14|P15|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P16|P17|P18|P19|P20|P21|P22|P23|P24|P25|P26|P27|P28|P29|P30|P31|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P32|P33|P34|P35|P36|P37|P38|P39|P40|P41|P42|P43|P44|P45|P46|P47|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P48|P49|P50|P51|P52|P53|P54|P55|P56|P57|P58|P59|P60|P61|P62|P63|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

In this format:

- o Type is set to ’1’ for SLLAO or ’2’ for TLLAO.
- o Length is set to the constant value ’5’ (i.e., 5 units of 8 octets).
- o X (proXy) is set to ’1’ in an S/TLLAO if the address corresponds to a Proxy; otherwise, X is set to ’0’.
- o Reserved is set to the value ’0’ on transmission and ignored on receipt.
- o Interface ID is set to a 16-bit integer value corresponding to an underlying interface of the AERO node. The value 255 is reserved for Server-based route optimization (see: Section 3.15.8).
- o UDP Port Number and IP Address are set to the addresses used by the AERO node when it sends encapsulated packets over the specified underlying interface (or to ’0’ when the addresses are left unspecified). When UDP is not used as part of the

encapsulation, UDP Port Number is set to '0'. When the encapsulation IP address family is IPv4, IP Address is formed as an IPv4-mapped IPv6 address as specified in Section 3.4.

- o P(i) is a set of 64 Preference values that correspond to the 64 Differentiated Service Code Point (DSCP) values [RFC2474]. Each P(i) is set to the value '0' ("disabled"), '1' ("low"), '2' ("medium") or '3' ("high") to indicate a QoS preference level for packet forwarding purposes.

AERO interfaces may be configured over multiple underlying interface connections to underlying links. 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, cellular, terrestrial, air-to-air directional, etc.) with diverse performance and cost properties.

A Client's underlying interfaces are classified as follows:

- o Native interfaces connect to the open Internetwork, and have a global IP address that is reachable from any open Internetwork correspondent.

- o NAT'ed interfaces connect to a closed network that is separated from the open Internetwork by a Network Address Translator (NAT). The NAT does not participate in any AERO control message signaling, but the AERO Server can issue AERO control messages on behalf of the Client.

- o VPN'ed interfaces use security encapsulation over the Internetwork to a Virtual Private Network (VPN) gateway that also acts as an AERO Server. As with NAT'ed links, the AERO Server can issue control messages on behalf of the Client.

- o Proxy'ed interfaces connect to a closed network that is separated from the open Internetwork by an AERO Proxy. Unlike NAT'ed and VPN'ed interfaces, the AERO Proxy (rather than the Server) can issue control message on behalf of the Client.

- o Direct interfaces connect the Client directly to a peer without crossing any networked paths. An example is a line-of-sight link between a remote pilot and an unmanned aircraft.

If a Client's multiple underlying interfaces are used "one at a time" (i.e., all other interfaces are in standby mode while one interface

is active), then ND messages include only a single S/TLLAO with Interface ID set to a constant value. In that case, the Client would appear to have a single underlying interface but with a dynamically changing link-layer address.

If the Client has multiple active underlying interfaces, then from the perspective of ND it would appear to have multiple link-layer addresses. In that case, ND messages MAY include multiple S/TLLAOs -- each with an Interface ID that corresponds to a specific underlying interface of the AERO node.

When the Client includes an S/TLLAO for an underlying interface for which it is aware that there is a NAT or Proxy on the path to the Server, or when a node includes an S/TLLAO solely for the purpose of announcing new QoS preferences, the node sets both UDP Port Number and IP Address to 0 to indicate that the addresses are unspecified.

When an ND message includes multiple S/TLLAOs, the first S/TLLAO MUST correspond to the AERO node's underlying interface used to transmit the message.

3.6. AERO Interface Initialization

3.6.1. AERO Relay Behavior

When a Relay enables an AERO interface, it first assigns an administratively-provisioned link-local address fe80::ID to the interface. Each fe80::ID address MUST be unique among all AERO nodes on the link. The Relay then engages in a dynamic routing protocol session with one or more Servers and all other Relays on the link (see: Section 3.3), and advertises its assigned ASPs into the native Internetwork.

Each Relay subsequently maintains an IP forwarding table entry for each active ACP covered by its ASP(s), and maintains neighbor cache entries for all Servers on the link. Relays exchange NS/NA messages with AERO link neighbors the same as for any AERO node. However, Neighbor Unreachability Detection (NUD) (see: Section 3.16) is optional since the dynamic routing protocol already provides reachability confirmation.

3.6.2. AERO Server Behavior

When a Server enables an AERO interface, it assigns an administratively-provisioned link-local address fe80::ID the same as for Relays. The Server further configures a service to facilitate PD exchanges with AERO Clients. The Server maintains neighbor cache entries for one or more Relays 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 their neighboring Relays (see: Section 3.3).

When the Server receives an NS/RS message from a Client on the AERO interface it authenticates the message and returns an NA/RA message. The Server further provides a simple link-layer conduit between AERO interface neighbors. In particular, when a packet sent by a source Client arrives on the Server's AERO interface and is destined to another AERO node, the Server forwards the packet from within the AERO interface driver at the link layer without ever disturbing the network layer.

3.6.3. AERO Client Behavior

When a Client enables an AERO interface, it sends RS messages with PD "Solicit" options over an underlying interface using the prefix-solicitation address as the source network layer address and all-routers [RFC4861] as the destination network layer address to obtain ACPs from one or more AERO Servers. Each Server processes the message and returns an RA message with a PD "Reply" option with the Server's link-layer address as the source and the base AERO address as the destination network layer addresses. In this way, the ND/PD control messages securely perform all autoconfiguration operations in a single request/response exchange.

After the initial ND/PD message exchange, the Client can register additional underlying interfaces with the Server by sending an RS message over each underlying interface using its base AERO address as the source network layer address and without including a PD option. The Server will update its neighbor cache entry for the Client and return an RA message.

The Client maintains a neighbor cache entry for each of its Servers and each of its active correspondent Clients. When the Client receives ND messages on the AERO interface it updates or creates neighbor cache entries, including link-layer address and QoS preferences.

3.6.4. AERO Proxy Behavior

When a Proxy enables an AERO interface, it maintains per-Client proxy neighbor cache entries based on control message exchanges. Proxies forward packets between their associated Clients and the Clients' associated Servers.

When the Proxy receives an RS message from a Client in the secured enclave, it creates an incomplete proxy neighbor cache entry and

forwards the message to a Server selected by the Client while using its own link-layer address as the source address. When the Server returns an RA message, the Proxy completes the proxy neighbor cache entry based on autoconfiguration information in the RA and forwards the RA to the Client while using its own link-layer address as the source address. The Client, Server and Proxy will then have the necessary state for managing the proxyed neighbor association.

3.7. 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", "proxy" 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 permanent neighbor cache entries for Servers on the link, and AERO Servers maintain permanent neighbor cache entries for Relays. 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 and maintained through ND/PD exchanges as specified in Section 3.14, and remain in place for durations bounded by ND/PD lifetimes. AERO Servers maintain static neighbor cache entries for each of their associated Clients, and AERO Clients maintain static neighbor cache entries for each of their associated Servers.

Proxy neighbor cache entries are created and maintained by AERO Proxies by gleaning information from Client/Server ND/PD exchanges, and remain in place for durations bounded by ND/PD lifetimes. AERO Proxies maintain proxy neighbor cache entries for each of their associated Clients, and include pointers to the Client’s current set of Servers.

Dynamic neighbor cache entries are created or updated based on receipt of route optimization messages as specified in Section 3.15, and are garbage-collected when keepalive timers expire. AERO nodes maintain dynamic neighbor cache entries for each of their active correspondents with lifetimes based on ND messaging constants.

When a target AERO node receives a valid NS message with an AERO source address, it returns an NA message and also creates or updates a dynamic neighbor cache entry for the source network-layer and link-layer addresses. 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. The node resets AcceptTime when it receives a new ND message, and otherwise decrements AcceptTime while no ND messages have been received. 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 route optimization procedure can converge before AcceptTime decrements below FORWARD_TIME (see below).

When a source AERO node receives a valid NA message with an AERO source address that matches its NS message, it creates or updates a dynamic neighbor cache entry for the target network-layer and link-layer addresses. The node then sets a "ForwardTime" variable in the neighbor cache entry to FORWARD_TIME seconds and uses this value to determine whether packets can be forwarded directly to the correspondent, i.e., instead of via a default route. The node resets ForwardTime when it receives a new NA, and otherwise decrements ForwardTime while no further NA messages have been received. It is RECOMMENDED that FORWARD_TIME be set to the default constant value 30 seconds to match the default REACHABLE_TIME value specified in [RFC4861].

The node also sets a "MaxRetry" variable to MAX_RETRY to limit the number of keepalives sent when a correspondent may have gone unreachable. It is RECOMMENDED that MAX_RETRY be set to 3 the same as described for address resolution in Section 7.3.3 of [RFC4861].

Different values for ACCEPT_TIME, FORWARD_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 route optimization procedure to converge.

When there may be a NAT between the Client and the Server, or if the path from the Client to the Server should be tested for reachability, the Client can send periodic RS messages to the Server without a PD option to receive RA replies. The RS/RA messaging will keep NAT state alive and test Server reachability without disturbing the PD service.

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

AERO interfaces may have multiple underlying interfaces and/or neighbor cache entries for neighbors with multiple Interface ID registrations (see Section 3.5). The AERO node uses each packet's DSCP value to select an outgoing underlying interface based on the node's own QoS preferences, and also to select a destination link-layer address based on the neighbor's underlying interface with the highest preference. If multiple outgoing interfaces and/or neighbor interfaces have a preference of "high", the AERO node sends one copy of the packet via each of the (outgoing / neighbor) interface pairs; otherwise, the node sends a single copy of the packet via the interface with the highest preference. AERO nodes keep track of which underlying interfaces are currently "reachable" or "unreachable", and only use "reachable" interfaces for forwarding purposes.

The following sections discuss the AERO interface forwarding algorithms for Clients, Proxies, Servers and Relays. In the following discussion, a packet's destination address is said to "match" if it is a non-link-local address with a prefix covered by an ASP/ACP, or if it is an AERO address that embeds an ACP, or if it is the same as an administratively-provisioned link-local address.

3.8.1. Client Forwarding Algorithm

When an IP packet enters a Client's AERO interface from the network layer the Client searches for a dynamic neighbor cache entry that matches the destination. If there is a match, the Client uses one or more "reachable" link-layer addresses in the entry as the link-layer addresses for encapsulation and admits the packet into the AERO link. Otherwise, the Client uses the link-layer address in a static neighbor cache entry for a Server as the encapsulation address (noting that there may be a Proxy on the path to the real Server).

When an IP packet enters a Client's AERO interface from the link-layer, if the destination matches one of the Client's ACPs or link-local addresses the Client decapsulates the packet and delivers it to the network layer. Otherwise, the Client drops the packet and MAY return a network-layer ICMP Destination Unreachable message subject to rate limiting (see: Section 3.13).

3.8.2. Proxy Forwarding Algorithm

When the Proxy receives a packet from a Client within the secured enclave, the Proxy searches for a dynamic neighbor cache entry that matches the destination. If there is a match, the Proxy uses one or more "reachable" link-layer addresses in the entry as the link-layer addresses for encapsulation and admits the packet into the AERO link. Otherwise, the Proxy uses the link-layer address for one of the Client's Servers as the encapsulation address.

When the Proxy receives a packet from an AERO interface neighbor, it searches for a proxy neighbor cache entry for a Client within the secured enclave that matches the destination. If there is a match, the Proxy forwards the packet to the Client. Otherwise, the Proxy returns the packet to the neighbor, i.e., by reversing the source and destination link-layer addresses.

3.8.3. Server Forwarding Algorithm

When an IP packet enters a Server's AERO interface from the network layer, the Server searches for a static neighbor cache entry for a Client that matches the destination. If there is a match, the Server uses one or more link-layer addresses in the entry as the link-layer addresses for encapsulation and admits the packet into the AERO link. Otherwise, the Server uses the link-layer address in a permanent neighbor cache entry for a Relay (selected through longest-prefix match) as the link-layer address for encapsulation.

When an IP packet enters a Server's AERO interface from the link layer, the Server processes the packet according to the network-layer destination address as follows:

- o if the destination matches one of the Server's own addresses the Server decapsulates the packet and forwards it to the network layer for local delivery.

- o else, if the destination matches a static neighbor cache entry for a Client the Server first determines whether the neighbor is the same as the one it received the packet from. If so, the Server drops the packet silently to avoid looping; otherwise, the Server uses the neighbor's link-layer address(es) as the destination for encapsulation and re-admits the packet into the AERO link.

- o else, the Server uses the link-layer address in a neighbor cache entry for a Relay (selected through longest-prefix match) as the link-layer address for encapsulation.

3.8.4. Relay Forwarding Algorithm

When an IP packet enters a Relay's AERO interface from the network layer, the Relay searches its IP forwarding table for an ACP entry that matches the destination and otherwise searches for a neighbor cache entry that matches the destination (e.g., for administratively-provisioned link-local addresses). If there is a match, the Relay uses the link-layer address in the corresponding neighbor cache entry as the link-layer address for encapsulation and forwards the packet into the AERO link. Otherwise, the Relay drops the packet and (for non-link-local addresses) returns a network-layer ICMP Destination Unreachable message subject to rate limiting (see: Section 3.13).

When an IP packet enters a Relay's AERO interface from the link-layer, the Relay processes the packet as follows:

- o if the destination does not match an ASP, or if the destination matches one of the Relay's own addresses, the Relay decapsulates the packet and forwards it to the network layer where it will be subject to either IP forwarding or local delivery.

- o else, if the destination matches an ACP entry in the IP forwarding table, or if the destination matches the link-local address in a permanent neighbor cache entry, the Relay first determines whether the neighbor is the same as the one it received the packet from. If so the Relay MUST drop the packet silently to avoid looping; otherwise, the Relay uses the neighbor's link-layer address as the destination for encapsulation and re-admits the packet into the AERO link.

- o else, the Relay drops the packet and (for non-link-local addresses) returns an ICMP Destination Unreachable message subject to rate limiting (see: Section 3.13).

3.8.5. Processing Return Packets

When an AERO node receives a return packet such as generated by an AERO Proxy (see Section 3.8.2), it proceeds according to the AERO link trust basis. Namely, the return packets have the same trust profile as for link-layer Destination Unreachable messages. If the node has sufficient trust basis to accept link-layer Destination Unreachable messages, it can then process the return packet as described in the following paragraph. Otherwise, the node SHOULD drop the packet and treat it as an indication that a path may be failing, and MAY use NUD to test the path for reachability.

If the node has sufficient trust basis to accept return packets, it searches for a dynamic neighbor cache entry that matches the

destination. If there is a match, the neighbor marks the corresponding link-layer address as "unreachable", selects the next-highest priority "reachable" link-layer address in the entry as the link-layer address for encapsulation then (re)admits the packet into the AERO link. If there are no "reachable" link-layer addresses, the neighbor instead sets FowardTime in the dynamic neighbor cache entry to 0. If the source address corresponds to one of the neighbor's own addresses, the neighbor also forwards the packet to the corresponding Server; otherwise, it drops the packet.

3.9. AERO Interface Encapsulation and Re-encapsulation

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

The AERO interface encapsulates packets per the Generic UDP Encapsulation (GUE) procedures in [I-D.ietf-intarea-gue][I-D.ietf-intarea-gue-extensions], or through an alternate encapsulation format (see: Appendix A). For packets entering the AERO interface from the network layer, the AERO interface copies the "TTL/Hop Limit", "Type of Service/Traffic Class" [RFC2983], "Flow Label"[RFC6438] (for IPv6) and "Congestion Experienced" [RFC3168] values in the packet's IP header into the corresponding fields in the encapsulation IP header. For packets undergoing re-encapsulation, the AERO interface instead copies these values from the original encapsulation IP header into the new encapsulation header, i.e., the values are transferred between encapsulation headers and *not* copied from the encapsulated packet's network-layer header. (Note especially that by copying the TTL/Hop Limit between encapsulation headers the value will eventually decrement to 0 if there is a (temporary) routing loop.) For IPv4 encapsulation/re-encapsulation, the AERO interface sets the DF bit as discussed in Section 3.12.

When GUE encapsulation is used, the AERO interface next sets the UDP source port to a constant value that it will use in each successive packet it sends, and sets the UDP length field to the length of the encapsulated packet plus 8 bytes for the UDP header itself plus the length of the GUE header (or 0 if GUE direct IP encapsulation is used). For packets sent to a Server or Relay, the AERO interface sets the UDP destination port to 8060, i.e., the IANA-registered port number for AERO. For packets sent to a Client, the AERO interface sets the UDP destination port to the port value stored in the neighbor cache entry for this Client. The AERO interface then either includes or omits the UDP checksum according to the GUE specification.

Clients normally use the IP address of the underlying interface as the encapsulation source address. If the underlying interface does not have an IP address, however, the Client uses an IP address taken from an ACP as the encapsulation source address (assuming the node has some way of injecting the ACP into the underlying network routing system). For IPv6 addresses, the Client normally uses the ACP Subnet Router Anycast address [RFC4291].

3.10. AERO Interface Decapsulation

AERO interfaces decapsulate packets destined either to the AERO node itself or to a destination reached via an interface other than the AERO interface the packet was received on. Decapsulation is per the procedures specified for the appropriate encapsulation format.

3.11. 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 ND messages from Clients, and create or update a static neighbor cache entry for the Client based on the specific message type.

- AERO Clients and Servers accept encapsulated packets if there is a static neighbor cache entry with a link-layer 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.

- AERO Proxies accept encapsulated packets if there is a proxy neighbor cache entry that matches the packet's network-layer destination address (i.e., the address of the Client) and link-layer source address (i.e., the address of one of the Client's Servers). When the proxy is configured to accept packets originating from any address in the open Internetwork however (e.g., from another Proxy), it omits the source address check.

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, e.g., as for common VPN systems such as OpenVPN [OVPN]. In environments where end systems use end-to-end security, however, it may be sufficient to require signatures only for ND and ICMP control plane messages and omit signatures for data plane messages.

3.12. AERO Interface Packet Size Issues

The AERO interface is the node's attachment to the AERO link. The AERO interface acts as a tunnel ingress when it sends a packet to an AERO link neighbor and as a tunnel egress when it receives a packet from an AERO link neighbor. AERO interfaces observe the packet sizing considerations for tunnels discussed in [I-D.ietf-intarea-tunnels] and as specified below.

The Internet Protocol expects that IP packets will either be delivered to the destination or a suitable Packet Too Big (PTB) message returned to support the process known as IP Path MTU Discovery (PMTUD) [RFC1191][RFC1981]. However, PTB messages may be crafted for malicious purposes such as denial of service, or lost in the network [RFC2923]. This can be especially problematic for tunnels, where a condition known as a PMTUD "black hole" can result. For these reasons, AERO interfaces employ operational procedures that avoid interactions with PMTUD, including the use of fragmentation when necessary.

AERO interfaces observe two different types of fragmentation. Source fragmentation occurs when the AERO interface (acting as a tunnel ingress) fragments the encapsulated packet into multiple fragments before admitting each fragment into the tunnel. Network fragmentation occurs when an encapsulated packet admitted into the tunnel by the ingress is fragmented by an IPv4 router on the path to the egress. Note that a packet that incurs source fragmentation may also incur network fragmentation.

IPv6 specifies a minimum link Maximum Transmission Unit (MTU) of 1280 bytes [RFC8200]. Although IPv4 specifies a smaller minimum link MTU of 68 bytes [RFC0791], AERO interfaces also observe the IPv6 minimum for IPv4 even if encapsulated packets may incur network fragmentation.

IPv6 specifies a minimum Maximum Reassembly Unit (MRU) of 1500 bytes [RFC8200], while the minimum MRU for IPv4 is only 576 bytes [RFC1122] (note that common IPv6 over IPv4 tunnels already assume a larger MRU than the IPv4 minimum).

AERO interfaces therefore configure an MTU that MUST NOT be smaller than 1280 bytes, MUST NOT be larger than the minimum MRU among all nodes on the AERO link minus the encapsulation overhead ("ENCAPS"), and SHOULD NOT be smaller than 1500 bytes. AERO interfaces also configure a Maximum Segment Unit (MSU) as the maximum-sized encapsulated packet that the ingress can inject into the tunnel without source fragmentation. The MSU value MUST NOT be larger than (MTU+ENCAPS) and MUST NOT be larger than 1280 bytes unless there is operational assurance that a larger size can traverse the link along all paths.

All AERO nodes MUST configure the same MTU/MSU values for reasons cited in [RFC3819][RFC4861]; in particular, multicast support requires a common MTU value among all nodes on the link. All AERO nodes MUST configure an MRU large enough to reassemble packets up to (MTU+ENCAPS) bytes in length; nodes that cannot configure a large-enough MRU MUST NOT enable an AERO interface.

The network layer proceeds as follow when it presents an IP packet to the AERO interface. For each IPv4 packet that is larger than the AERO interface MTU and with the DF bit set to 0, the network layer uses IPv4 fragmentation to break the packet into a minimum number of non-overlapping fragments where the first fragment is no larger than the MTU and the remaining fragments are no larger than the first. For all other IP packets, if the packet is larger than the AERO interface MTU, the network layer drops the packet and returns a PTB message to the original source. Otherwise, the network layer admits each IP packet or fragment into the AERO interface.

For each IP packet admitted into the AERO interface, the interface (acting as a tunnel ingress) encapsulates the packet. If the encapsulated packet is larger than the AERO interface MSU the ingress source-fragments the encapsulated packet into a minimum number of non-overlapping fragments where the first fragment is no larger than the MSU and the remaining fragments are no larger than the first. The ingress then admits each encapsulated packet or fragment into the tunnel, and for IPv4 sets the DF bit to 0 in the IP encapsulation header in case any network fragmentation is necessary. The encapsulated packets will be delivered to the egress, which reassembles them into a whole packet if necessary.

Several factors must be considered when fragmentation is needed. For AERO links over IPv4, the IP ID field is only 16 bits in length, meaning that fragmentation at high data rates could result in data corruption due to reassembly misassociations [RFC6864][RFC4963]. 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]. In environments where IP fragmentation

issues could result in operational problems, the ingress SHOULD employ intermediate-layer source fragmentation (see: [RFC2764] and [I-D.ietf-intarea-gue-extensions]) before appending the outer encapsulation headers to each fragment. Since the encapsulation fragment header reduces the room available for packet data, but the original source has no way to control its insertion, the ingress MUST include the fragment header length in the ENCAPS length even for packets in which the header is absent.

3.13. AERO Interface Error Handling

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

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

The IP header is followed by an ICMP header that includes an error Type, Code and Checksum. Valid type values include "Destination Unreachable", "Time Exceeded" and "Parameter Problem" [RFC0792][RFC4443]. (AERO interfaces ignore all link-layer IPv4 "Fragmentation Needed" and IPv6 "Packet Too Big" messages since they only emit packets that are guaranteed to be no larger than the IP minimum link MTU as discussed in Section 3.12.)

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

The link-layer error message format is shown in Figure 3 (where, "L2" and "L3" refer to link-layer and network-layer, respectively):

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

Figure 3: AERO Interface Link-Layer Error Message Format

The AERO node rules for processing these link-layer error messages are as follows:

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

- o When an AERO node receives persistent link-layer Time Exceeded messages, the IP ID field may be wrapping before earlier fragments awaiting reassembly have been processed. In that case, the node SHOULD begin including integrity checks and/or institute rate limits for subsequent packets.

- o When an AERO node receives persistent link-layer Destination Unreachable messages in response to encapsulated packets that it sends to one of its dynamic neighbor correspondents, the node SHOULD process the message as an indication that a path may be failing, and MAY initiate NUD over that path. If it receives Destination Unreachable messages on many or all paths, the node SHOULD set ForwardTime for the corresponding dynamic neighbor

cache entry to 0 and allow future packets destined to the correspondent to flow through a default route.

- o When an AERO Client receives persistent link-layer Destination Unreachable messages in response to encapsulated packets that it sends to one of its static neighbor Servers, the Client SHOULD math the path as unusable and use another path. If it receives Destination Unreachable messages on many or all paths, the Client SHOULD associate with a new Server and send a PD "Release" message to the old Server as specified in Section 3.17.6.

- o When an AERO Server receives persistent link-layer Destination Unreachable messages in response to encapsulated packets that it sends to one of its static neighbor Clients, the Server SHOULD mark the path as unusable and use another path. If it receives Destination Unreachable messages on multiple paths, the Server should take no further actions unless it receives a PD "Release" message or if the PD lifetime expires. In that case, the Server MUST release the Client’s delegated ACP, withdraw the ACP from the AERO routing system and delete the neighbor cache entry.

- o When an AERO Relay or Server receives link-layer Destination Unreachable messages in response to an encapsulated packet that it sends to one of its permanent neighbors, it treats the messages as an indication that the path to the neighbor may be failing. However, the dynamic routing protocol should soon reconverge and correct the temporary outage.

When an AERO Relay receives a packet for which the network-layer 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 a network-layer Destination Unreachable message subject to rate limiting. The Relay first writes the network-layer source address of the original packet as the destination address of the 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 source address of the message, then encapsulates the message and forwards it 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 message and forwards it via a link outside the AERO interface.

When an AERO node receives an encapsulated packet for which the reassembly buffer it too small, it drops the packet and returns a network-layer Packet Too Big (PTB) message. The node first writes the MRU value into the PTB message MTU field, writes the network-layer source address of the original packet as the destination address of the 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 source address of the message, then encapsulates the message and forwards it to the next hop within the AERO interface. Otherwise, the node uses one of its non link-local addresses as the source address of the message and forwards it via a link outside the AERO interface.

When an AERO node receives any network-layer error message via the AERO interface, it examines the network-layer destination address. If the next hop toward the destination is via the AERO interface, the node re-encapsulates and forwards the message to the next hop within the AERO interface. Otherwise, if the network-layer source address 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, recalculates the IP and/or ICMP checksums then forwards the message via a link outside the AERO interface.

3.14. AERO Router Discovery, Prefix Delegation and Autoconfiguration

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

3.14.1. AERO ND/PD Service Model

Each AERO Server configures a PD service to facilitate Client requests. Each Server is provisioned with a database of ACP-to-Client ID mappings for all Clients enrolled in the AERO system, as well as any information necessary to authenticate each Client. The Client database is maintained by a central administrative authority for the AERO link and securely distributed to all Servers, e.g., via the Lightweight Directory Access Protocol (LDAP) [RFC4511], via static configuration, etc. Therefore, no Server-to-Server PD state synchronization is necessary, and Clients can optionally hold separate PDs for the same ACPs from multiple Servers. In this way, Clients can associate with multiple Servers, and can receive new PDs from new Servers before releasing PDs received from existing Servers. This provides the Client with a natural fault-tolerance and/or load balancing profile.

AERO Clients and Servers use ND messages to maintain neighbor cache entries. AERO Servers configure their AERO interfaces as advertising interfaces, and therefore send unicast RA messages with configuration information in response to a Client’s RS message. The RS/RA messaging is conducted in the same fashion as specified in [RFC5214].

AERO Clients and Servers include PD messages as options in the RS/RA messages they exchange (see: [I-D.templin-6man-dhcpv6-ndopt]).

Client-initiated PD options are included in RS messages, and Server-initiated PD options are included in RA messages. The unified ND/PD messages are exchanged between Client and Server according to the prefix management schedule determined by the PD service. The unified messages can be protected using SEcure Neighbor Discovery (SEND) [RFC3971].

On Some AERO links, PD arrangements may be through some out-of-band service such as network management, static configuration, etc. In those cases, AERO nodes can use simple RS/RA message exchanges with no explicit PD options. Instead, the RS/RA messages use AERO addresses as a means of representing the delegated prefixes, e.g., if a message includes a source address of "fe80::2001:db8:1:2" then the recipient can infer that the sender holds the prefix delegation "2001:db8:1:2::/64".

The following sections specify the Client and Server behavior.

3.14.2. AERO Client Behavior

AERO Clients discover the link-layer addresses of AERO Servers via static configuration (e.g., from a flat-file map of Server addresses and locations), or through an automated means such as Domain Name System (DNS) name resolution [RFC1035]. In the absence of other information, the Client resolves the DNS Fully-Qualfied Domain Name (FQDN) "linkupnetworks.[domainname]" where "linkupnetworks" is a constant text string and "[domainname]" is a DNS suffix for the Client’s underlying interface (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 ACPs through a combined ND/PD message exchange. The Client includes a PD "Solicit" message as an ND option in an RS message with the prefix-solicitation address as the IPv6 source address, all-routers multicast as the IPv6 destination address, the address of the Client’s underlying interface as the link-layer source address and the link-layer address of the Server as the link-layer destination address. (If the Client’s underlying interface does not have an IP address, the Client can use the ACP Subnet Router Anycast address as the link-layer source address.)

The Client next includes a "Client Identifier" and an "IA_PD" (i.e., prefix request) code in the PD "Solicit" message. If the Client is pre-provisioned with ACPs associated with the AERO service, it MAY also include the ACPs in the "IA_PD" option to indicate its preferences to the Server. The Client finally includes any additional PD codes (e.g., "Rapid Commit").

The Client next includes one or more SLLAOs in the RS message formatted as described in Section 3.5 to register its link-layer address(es) with the Server. The first SLLAO MUST correspond to the underlying interface over which the Client will send the RS message. The Client MAY include additional SLLAOs specific to other underlying interfaces, but if so it MUST have assurance that there will be no NATs or Proxies on the paths to the Server via those interfaces. (Otherwise, the Client can register additional link-layer addresses with the Server by sending subsequent NS/RS messages via different underlying interfaces after the initial RS/RA exchange).

The Client then sends the RS message to the AERO Server and waits for an RA message reply (see Section 3.14.3) while retrying MAX_RETRY times until an RA is received. If no RA is received, or if it receives an RA with Router Lifetime set to 0 and/or a "Reply" with no ACPs, the Client SHOULD discontinue autoconfiguration attempts through this Server and try another Server. Otherwise, the Client processes the ACPs in the embedded "Reply" message.

Next, the Client creates a static neighbor cache entry with the Server’s link-local address as the network-layer address and the Server’s encapsulation source address as the link-layer address. The Client then autoconfigures AERO addresses for each of the delegated ACPs and assigns them to the AERO interface.

The Client next examines the P bit in the RA message flags field [RFC5175]. If the P bit value was 1, the Client assumes that there is a NAT or Proxy on the path to the Server via the interface over which it sent the RS message. In that case, the Client sets UDP Port Number and IP Address to 0 in the S/TLLAOs of any subsequent ND messages it sends to the Server over that link.

The Client also caches any ASPs included in Route Information Options (RIOs) [RFC4191] as ASPs to associate with the AERO link, and assigns the MTU/MSU values in the MTU options to its AERO interface while configuring an appropriate MRU. This configuration information applies to the AERO link as a whole, and all AERO nodes will receive the same values.

Following autoconfiguration, the Client sub-delegates the ACPs to its attached EUNs and/or the Client’s own internal virtual interfaces as described in [I-D.templin-v6ops-pdhost]. The Client subsequently maintains its ACP delegations through each of its Servers by sending RS "Renew", "Rebind", and/or "Release" messages. The Server will in turn send RA "Reply" messages.

After the Client registers its Interface IDs and their associated UDP/IP addresses and ’P(i)’ values, it may wish to change one or more

Interface ID registrations, e.g., if an underlying interface changes address or becomes unavailable, if QoS preferences change, etc. To do so, the Client prepares an unsolicited NA message to send over any available underlying interface. The source and target address of the NA message are set to the Client's AERO address, and the destination address is set to all-nodes multicast. The NA MUST include a TLLAO specific to the selected available underlying interface as the first TLLAO and MAY include any additional TLLAOs specific to other underlying interfaces. The Client includes fresh 'P(i)' values in each TLLAO to update the Server's neighbor cache entry. If the Client wishes to update 'P(i)' values without updating the link-layer address, it sets the UDP Port Number and IP Address fields to 0. If the Client wishes to disable the interface, it sets all 'P(i)' values to '0' ("disabled").

If the Client wishes to discontinue use of a Server it issues an RS "Release" message. When the Server processes the message, it releases the ACP, deletes its neighbor cache entry for the Client, withdraws the IP route from the routing system and returns an RA "Reply".

3.14.3. AERO Server Behavior

AERO Servers act as IPv6 routers and support a PD service on their AERO links. AERO Servers arrange to add their encapsulation layer IP addresses (i.e., their link-layer addresses) to a static map of Server addresses for the link and/or the DNS resource records for the FQDN "linkupnetworks.[domainname]" before entering service.

When an AERO Server receives a prospective Client's RS "Solicit" message on its AERO interface, and the Server is too busy, it SHOULD return an immediate RA "Reply" message with no ACPs and with Router Lifetime set to 0. Otherwise, the Server authenticates the RS message and processes the embedded "Solicit" option. The Server first determines the correct ACPs to delegate to the Client by searching the Client database. When the Server delegates the ACPs, it also creates an IP forwarding table entry for each ACP so that the AERO BGP-based routing system will propagate the ACPs to the Relays that aggregate the corresponding ASP (see: Section 3.3).

Next, the Server prepares an RA "Reply" message that includes the delegated ACPs. For IPv4 ACPs, the ACP is in IPv4-mapped IPv6 address format and with prefix length set as specified in Section 3.4. The Server then prepares an RA "Reply" message using its link-local address (i.e., fe80::ID) as the network-layer source address, the Client's base AERO address from the first ACP as the network-layer destination address, the Server's link-layer address as the source link-layer address, and the source link-layer address of

the RS message as the destination link-layer address. The Server next sets the P flag in the RA message flags field [RFC5175] to 1 if the source link-layer address in the RS message was different than the address in the first SLLAO to indicate that there is a NAT or Proxy on the path; otherwise it sets P to 0. The Server then includes one or more RIOs that encode the ASPs for the AERO link. The Server also includes two MTU options - the first MTU option includes the MTU for the link and the second MTU option includes the MSU for the link (see Section 3.12). The Server finally sends the RA "Reply" message to the Client.

The Server next creates a static neighbor cache entry for the Client using the base AERO address as the network-layer address and with lifetime set to no more than the smallest PD lifetime. Next, the Server updates the neighbor cache entry link-layer address(es) by recording the information in each SLLAO option indexed by the Interface ID and including the UDP port number, IP address and P(i) values. For the first SLLAO in the list, however, the Server records the actual encapsulation source UDP and IP addresses instead of those that appear in the SLLAO in case there was a NAT or Proxy in the path.

After the initial RS/RA exchange, the AERO Server maintains the neighbor cache entry for the Client until the PD lifetimes expire. If the Client issues an RS "Renew", the Server extends the PD lifetimes. If the Client issues an RS "Release", or if the Client does not issue a "Renew" before the lifetime expires, the Server deletes the neighbor cache entry for the Client and withdraws the IP routes from the AERO routing system. The Server processes these and any other Client PD messages, and returns an RA "Reply". The Server may also issue an unsolicited RA "Reconfigure" message to inform the Client that it needs to renegotiate its PDs.

3.14.3.1. Lightweight DHCPv6 Relay Agent (LDRA)

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

When the LDRA receives an authentic RS message, it extracts the PD message option and wraps it in IPv6/UDP headers. It sets the IPv6 source address to the source address of the RS message, sets the IPv6 destination address to 'All_DHCP_Relay_Agents_and_Servers' and sets

the UDP fields to values that will be understood by the DHCPv6 server.

The LDRA then wraps the message in a Relay-Forward message header and includes an Interface-ID option that includes enough information to allow the LDRA to forward the resulting Reply message back to the Client (e.g., the Client's link-layer addresses, a security association identifier, etc.). The LDRA also wraps the information in all of the SLLAO options from the RS message into the Interface-ID option, then forwards the message to the DHCPv6 server.

When the DHCPv6 server prepares a Reply message, it wraps the message in a Relay-Reply message and echoes the Interface-ID option. The DHCPv6 server then delivers the Relay-Reply message to the LDRA, which discards the Relay-Reply wrapper and IPv6/UDP headers, then delivers the DHCPv6 message to be wrapped into an RA response to the Client. The Server uses the information in the Interface ID option to prepare the RA message and to cache the link-layer addresses taken from the SLLAOs echoed in the Interface-ID option.

3.15. AERO Interface 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 MAY initiate an AERO link route optimization procedure on behalf of any of its native underlying interfaces. The procedure is based on an exchange of IPv6 ND messages using a chain of AERO Servers and Relays as a trust basis.

Although the Client is responsible for initiating route optimization, the Server is the policy enforcement point that determines whether route optimization is permitted. For example, on some AERO links route optimization would allow traffic to circumvent critical network-based traffic inspection points. In those cases, the Server can simply discard any route optimization messages instead of forwarding them.

The following sections specify the AERO link route optimization procedure.

3.15.1. Reference Operational Scenario

Figure 4 depicts the AERO link 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(C1)                                          L2(C2)
 fe80::2001:db8:0:0                                fe80::2001:db8:1:0
  +--------------+                                  +--------------+
  |AERO Client C1|                                  |AERO Client C2|
  +--------------+                                  +--------------+
  2001:DB8:0::/48                                   2001:DB8:1::/48
          |                                                 |
         .-.                                               .-.
      ,-(  _)-.   2001:db8:0::1      2001:db8:1::1     ,-(  _)-.
   .-(_  IP   )-.   +-------+          +-------+    .-(_  IP   )-.
  (__     EUN      )--|Host H1|          |Host H2|--(__     EUN      )
     `-(______)-'    +-------+          +-------+     `-(______)-'
```

Figure 4: AERO Reference Operational Scenario

In Figure 4, Relay ('R1') assigns the administratively-provisioned link-local 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 an ND/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 an ND/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 ('H2') connects to the EUN, and configures the address 2001:db8:1::1.

3.15.2. Concept of Operations

Again, with reference to Figure 4, 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 an NS message toward Client ('C2') via Server ('S1').

Server ('S1') then re-encapsulates and forwards both the packet and the NS message out the same AERO interface toward Client ('C2') via Relay ('R1'). When Relay ('R1') receives the packet and NS 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 NS message to Server ('S2'), which then forwards them to Client ('C2').

After Client ('C2') receives the NS message, it process the message and creates or updates a dynamic neighbor cache entry for Client ('C1'), then sends the NA response to the link-layer address of Server ('S2'). When Server ('S2') receives the NA message it re-encapsulates the message and forwards it on to Relay ('R1'), which re-encapsulates and forwards the message on to Server ('S1') which re-encapsulates and forwards the message on to Client ('C1').

After Client ('C1') receives the NA message, it processes the message and creates or updates a dynamic neighbor cache entry for Client ('C2'). Thereafter, 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.

3.15.3. Sending NS Messages

When a Client forwards a packet with a source address from one of its ACPs toward a destination address covered by an ASP (i.e., toward another AERO Client connected to the same AERO link), the source Client MAY send an NS 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 an NS message forward toward Client ('C2'), subject to rate limiting (see Section 8.2 of [RFC4861]). Client ('C1') prepares the NS message as follows:

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

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

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

- o the network-layer destination address is set to the AERO address corresponding to the destination address of Client ('C2').

- o the Type is set to 135.

- o the Target Address is set to the destination address of the packet that triggered route optimization.

- o the message includes one or more SLLAOs set to appropriate values for Client ('C1')'s native underlying interfaces.

- o the message includes one or more RIOs that include Client ('C1')'s ACPs [I-D.templin-6man-rio-redirect].

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

Note that the act of sending NS 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').

3.15.4. Re-encapsulating and Relaying the NS

When Server ('S1') receives an NS message from Client ('C1'), it first verifies that the SLLAOs in the NS are a proper subset of the link-layer addresses in Client ('C1')'s neighbor cache entry. If the Client's SLLAOs are not acceptable, Server ('S1') discards the message. Otherwise, Server ('S1') verifies that Client ('C1') is authorized to use the ACPs encoded in the RIOs of the NS and discards the NS if verification fails.

Server ('S1') then examines the network-layer destination address of the NS 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 NS 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 NS 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 NS 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 NS via Server ('S2').

When Server ('S2') receives the NS message from Relay ('R1') it determines that Client ('C2') is a neighbor by consulting its neighbor cache. Server ('S2') then re-encapsulates the NS 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').

3.15.5. Processing NSs and Sending NAs

When Client ('C2') receives the NS message, it accepts the NS 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 route optimization target.

In the reference operational scenario, when Client ('C2') receives a valid NS message, it either creates or updates a dynamic neighbor cache entry that stores the source address of the message as the network-layer address of Client ('C1') , stores the link-layer addresses found in the SLLAOs as the link-layer addresses of Client ('C1'), and stores the ACPs encoded in the RIOs of the NS as the ACPs for Client ('C1'). Client ('C2') then sets AcceptTime for the neighbor cache entry to ACCEPT_TIME.

After processing the message, Client ('C2') prepares an NA message response as follows:

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

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

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

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

- o the Type is set to 136.

- o The Target Address is set to the Target Address field in the NS message.

- o the message includes one or more TLLAOs set to appropriate values for Client ('C2')'s native underlying interfaces.

- o the message includes one or more RIOs that include Client ('C2')'s ACPs [I-D.templin-6man-rio-redirect].

- o the message SHOULD include a Timestamp option and MUST echo the Nonce option received in the NS (i.e., if a Nonce option is included).

Client ('C2') then sends the NA message to Server ('S2').

3.15.6. Re-encapsulating and Relaying NAs

When Server ('S2') receives an NA message from Client ('C2'), it first verifies that the TLLAOs in the NA are a proper subset of the Interface IDs in Client ('C2')'s neighbor cache entry. If the Client's TLLAOs are not acceptable, Server ('S2') discards the message. Otherwise, Server ('S2') verifies that Client ('C2') is authorized to use the ACPs encoded in the RIOs of the NA message. If validation fails, Server ('S2') discards the NA.

Server ('S2') then examines the network-layer destination address of the NA to determine the next hop toward Client ('C1') by searching for the AERO address in the neighbor cache. Since Client ('C1') is not a neighbor, Server ('S2') re-encapsulates the NA 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 NA 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 NA 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 NA via Server ('S1').

When Server ('S1') receives the NA message from Relay ('R1') it determines that Client ('C1') is a neighbor by consulting its neighbor cache. Server ('S1') then re-encapsulates the NA 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').

3.15.7. Processing NAs

When Client ('C1') receives the NA message, it first verifies the Nonce value matches the value that it included in its NS message (if any). If the Nonce values match, Client ('C1') then processes the message as follows.

In the reference operational scenario, when Client ('C1') receives the NA message, it either creates or updates a dynamic neighbor cache entry that stores the source 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 ACPs encoded in the RIOs of the NA as the ACPs for Client ('C2'). 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 NS/NA message exchange is required in the reverse direction; hence, the mechanism is asymmetric.)

3.15.8. Server and Proxy Extended Route Optimization

Route optimization may be initiated by the source Client by sending NS messages with SLLAOs corresponding to its native underlying interfaces. Route optimization for the source Client's other interfaces may be initiated by Servers and/or Proxies. Each node initiates route optimization by sending NS messages with SLLAOs only for those underlying interfaces they are authoritative for. Each node MUST consistently use the same Interface ID values to denote the same interfaces. The Interface IDs are established and maintained by the source Client's RS/RA exchanges.

The target Client's Server serves as a route optimization target if some or all of the target Client's underlying interfaces connect via NATs, Proxies and/or VPNs. In that case, when the source sends an NS message the target Server both forwards the NS toward a native underlying interface of the target Client (if any) and prepares an NA response the same as if it were the target Client (see: Section 3.15.5). (This means that the source may receive two

separate NA messages - one from the target Server and one from the target Client. The source must accept the union of the information from both messages.)

For non-native underlying interfaces, the target Server includes a first TLLAO option in the NA with Interface ID set to 255 and includes any additional TLLAOs corresponding to the Client's NATed, Proxyed and/or VPNed underlying interfaces. The Server writes its own link-layer address in TLLAOs corresponding to NATed and VPNed underlying interfaces, and writes the link-layer address of the Proxy in TLLAOs corresponding to Proxyed underlying interfaces (while also setting the X flag). The Interface ID and QoS Preference values in the TLLAOs are those supplied by the Client during the initial RS/RA exchange and updated by any ensuing unsolicited NA messages. The target Server must then maintain a dynamic neighbor cache entry for the Client, but MUST NOT send BGP updates for Clients discovered through dynamic route optimization.

Thereafter, if the target Client moves to a new Server, the old Server sends unsolicited NA messages with no TLLAOs (subject to rate limiting) back to the source in response to data packets received from a correspondent node while forwarding the packets themselves to a Relay. The Relay will then either forward the packets to the new Server if the target Client has moved, or drop the packets if the target Client is no longer in the network. The source then allows future packets destined to the target Client to again flow through its own Server (or Relay). Note however that the old Server retains the neighbor cache entry with its associated AcceptTime since there may be many packets in flight. AcceptTime will then eventually decrement to 0 once the correspondent node processes and acts on the unsolicited NAs.

When the target Client (or Proxy) sends unsolicited NA messages to the target Server to update link-layer address and/or QoS preferences, the target Server repeats the messages to any of its dynamic neighbors while using its own link-layer and link-local addresses as the source addresses. In this way, the target Server acts as a link-scoped multicast repeater on behalf of the target Client (or Proxy).

(Note that instead of serving as the route optimization target for Proxy interfaces, the target Server could instead forward the source's NS messages and allow the Proxies to return NA messages, i.e., the same as for Clients on native interfaces. That would mean that the source could receive multiple NA messages from multiple Proxies and, if some or all NA messages are lost, the source would not be able to determine the full picture of the Client's Proxy affiliations. If this alternate architecture is deemed appropriate

in some use cases, then the AERO Proxies could be employed to serve as route optimization targets instead of depending on the Servers to do so.)

3.16. Neighbor Unreachability Detection (NUD)

AERO nodes perform Neighbor Unreachability Detection (NUD) by sending NS messages to elicit solicited NA messages from neighbors the same as described in [RFC4861]. NUD is performed either reactively in response to persistent link-layer errors (see Section 3.13) or proactively to update neighbor cache entry timers and/or link-layer address information.

When an AERO node sends an NS/NA message, it uses one of its link-local addresses as the IPv6 source address and a link-local address of the neighbor as the IPv6 destination address. When route optimization directs a source AERO node to a target AERO node, the source node SHOULD proactively test the direct path by sending an initial NS message to elicit a solicited NA response. While testing the path, the source node can optionally continue sending packets via its default router, maintain a small queue of packets until target reachability is confirmed, or (optimistically) allow packets to flow directly to the target.

While data packets are still flowing, the source node thereafter periodically tests the direct path to the target node (see Section 7.3 of [RFC4861]) in order to keep dynamic neighbor cache entries alive. When the target node receives a valid NS message, it resets AcceptTime to ACCEPT_TIME and updates its cached link-layer addresses (if necessary). When the source node receives a solicited NA message, it resets ForwardTime to FORWARD_TIME and updates its cached link-layer addresses (if necessary). If the source node is unable to elicit a solicited NA response from the target node after MaxRetry attempts, it SHOULD set ForwardTime to 0. Otherwise, the source node considers the path usable and SHOULD thereafter process any link-layer errors as an indication that the direct path to the target node has either failed or has become intermittent.

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

Note that an AERO node may have multiple underlying interface paths toward the target neighbor. In that case, the node SHOULD perform NUD over each underlying interface and only consider the neighbor unreachable if NUD fails over multiple underlying interface paths.

3.17. Mobility Management and Quality of Service (QoS)

AERO is an example of a Distributed Mobility Management (DMM) service. Each AERO Server is responsible for only a subset of the Clients on the AERO link, as opposed to a Centralized Mobility Management (CMM) service where there is a single network service for all Clients. AERO Clients coordinate with their regional Servers via RS/RA exchanges to maintain the DMM profile, and the AERO routing system tracks the current AERO Client/Server peering relationships.

Mobility management for AERO interfaces is accommodated by sending unsolicited NA messages the same as for announcing link-layer address changes for any interface that implements IPv6 ND [RFC4861]. When a node sends an unsolicited NA message, it sets the IPv6 source to its own link-local address, sets the IPv6 destination address to all-nodes multicast, sets the link-layer source address to its own address and sets the link-layer destination address to either a multicast address or the unicast link-layer address of a neighbor. If the unsolicited NA message must be received by multiple neighbors, the node sends multiple copies of the NA using a different unicast link-layer destination address for each neighbor. Mobility management considerations are specified in the following sections.

3.17.1. Forwarding Packets on Behalf of Departed Clients

When a Server receives packets with destination addresses that do not match one of its static neighbor cache Clients, it forwards the packets to a Relay and also returns an unsolicited NA message to the sender with no TLLAOs. The packets will be delivered to the target Client’s new location, and the sender will realize that it needs to deprecate its routing information that associated the target with this Server.

3.17.2. Announcing Link-Layer Address and QoS Preference Changes

When a Client needs to change its link-layer addresses, e.g., due to a mobility event, it sends unsolicited NAs to its neighbors using the new link-layer address as the source address and with TLLAOs that include the new Client UDP Port Number, IP Address and P(i) values. If the Client sends the NA solely for the purpose of updating QoS preferences without updating the link-layer address, the Client sets the UDP Port Number and IP Address to 0.

The Client MAY send up to MaxRetry unsolicited NA messages in parallel with sending actual data packets in case one or more NAs are lost. If all NAs are lost, the neighbor will eventually invoke NUD by sending NS messages that include SLLAOs.

3.17.3. Bringing New Links Into Service

When a Client needs to bring new underlying interfaces into service (e.g., when it activates a new data link), it sends unsolicited NAs to its neighbors using the new link-layer address as the source address and with TLLAOs that include the new Client link-layer information.

3.17.4. Removing Existing Links from Service

When a Client needs to remove existing underlying interfaces from service (e.g., when it de-activates an existing data link), it sends unsolicited NAs to its neighbors with TLLAOs with all P(i) values set to 0.

If the Client needs to send the unsolicited NAs over an underlying interface other than the one being removed from service, it MUST include a current TLLAO for the sending interface as the first TLLAO and include TLLAOs for any underlying interface being removed from service as additional TLLAOs.

3.17.5. Implicit Mobility Management

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

If the Client’s underlying interface address changes (either due to a readdressing of the original interface or switching to a new interface) the neighbor immediately updates the neighbor cache entry for the Client and begins accepting and sending packets to the Client’s new link-layer address. This implicit mobility method applies to use cases such as cellphones with both WiFi and Cellular interfaces where only one of the interfaces is active at a given time, and the Client automatically switches over to the backup interface if the primary interface fails.

3.17.6. Moving to a New Server

When a Client associates with a new Server, it performs the Client procedures specified in Section 3.14.2.

When a Client disassociates with an existing Server, it sends an RS "Release" message via a new Server with its base AERO address as the network-layer source address and the (administratively-provisioned) link-local address of the old Server as the network-layer destination address. The new Server then caches the Client's AERO address and "Release" message parameters (e.g., "transaction ID") and writes its own administratively-provisioned link-local address as the network-layer source address. The new Server then forwards the message to a Relay, which forwards the message to the old Server.

When the old Server receives the "Release", it releases the Client's ACP prefix delegations and routes. The old Server then deletes the Client's neighbor cache entry so that any in-flight packets will be forwarded via a Relay to the new Server, which will forward them to the Client. The old Server finally returns a "Reply" message via a Relay to the new Server, which will decapsulate the "Reply" message and forward it as an RA "Reply" to the Client.

When the new Server forwards the "Reply" message, the Client can delete both the default route and the neighbor cache entry for the old Server. (Note that since messages may be lost in the network the Client SHOULD retry until it gets an RA "Reply" indicating that the RS "Release" was successful. If the Client does not receive a "Reply" after MaxRetry attempts, the old Server may have failed and the Client should discontinue its "Release" attempts.)

Finally, 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.

3.18. Multicast Considerations

When the underlying network does not support multicast, AERO Clients map link-scoped multicast addresses to the link-layer address of a Server, which acts as a multicast forwarding agent. The AERO Client also serves as an IGMP/MLD Proxy for its EUNs and/or hosted applications per [RFC4605] while using the link-layer address of the Server as the link-layer address for all multicast packets.

When the underlying network supports multicast, AERO nodes use the multicast address mapping specification found in [RFC2529] for IPv4 underlying networks and use a TBD site-scoped multicast mapping for IPv6 underlying networks. In that case, border routers must ensure that the encapsulated site-scoped multicast packets do not leak outside of the site spanned by the AERO link.

4. The AERO Proxy

In some deployments, AERO Clients may be located in secured enclaves (e.g., a corporate enterprise network, a radio access network, etc.) that do not allow direct communications from the Client to a Server in the outside Internetwork. In that case, the secured enclave can employ an AERO Proxy.

The AERO Proxy is located at the secured enclave perimeter and listens for RS messages originating from or RA messages destined to AERO Clients located within the enclave. The Proxy acts on these control messages as follows:

- when the Proxy receives an RS message from a Client within the secured enclave, it first authenticates the message then creates a proxy neighbor cache entry for the Client in the INCOMPLETE State and caches the Client and Server link-layer address along with any identifying information including PD "transaction IDs", "Client Identifiers", etc. and/or ND Nonce values. The Proxy then re-encapsulates the message and forwards it to the Server indicated by the destination link-layer address in the packet while substituting its own external address as the source link-layer address.

- when the Proxy receives an RA message from the Server, it matches the message with the (INCOMPLETE) proxy neighbor cache entry. The Proxy then caches the route information in the message as a mapping from the Client’s ACPs to the Client’s address within the secured enclave, and sets the neighbor cache entry state to REACHABLE. The Proxy then re-encapsulates the message and forwards it to the Client. At the same time, the Proxy sends an unsolicited NA message including a TLLAO with the X flag set back to the Server to assert that it is indeed a Proxy as opposed to an ordinary NAT. (In environments where spoofing is a threat, the Proxy signs the NA using SEND.)

After the initial RS/RA handshake, the Proxy can send unsolicited NA messages to the Client’s Server(s) to update Server neighbor cache entries on behalf of the Client. (For example, the Proxy can send NA messages with a TLLAO with UDP Port Number and IP Address set to 0 and with valid P(i) values to update the Server(s) with the Client’s

new QoS preferences for that link). The Proxy also forwards any unsolicited NA messages originating from the Client to the Client's Server(s) (e.g. if the Client needs to announce new QoS preferences on its own behalf), and forwards any data packets originating from the Client to the Client's primary Server.

At the same time, for data packets originating from a Client within the enclave with destination addresses that match an ASP, the Proxy can initiate route optimization by sending an NS message via the Server to solicit an NA message from a target node on the path to the destination Client the same as discussed in Section 3.15. The target must deliver the NA message directly to the Proxy, i.e., instead of relaying through the backward chain of Relays and Servers, since the backward chain could deliver the NA to a different Proxy besides the one that produced the NS. For this reason, the Proxy prepares an NS message as specified in Section 3.15.3, but with its own link-layer address as the link-layer source address and with a single SLLAO containing its link-layer address and with the X flag set to indicate that direct delivery is required.

When the target receives the NS message, it creates a dynamic neighbor cache entry in the ACCEPT state and returns an NA message directly to the Proxy. When the target is a Client, it includes TLLAOs in the NA message with link-layer addresses corresponding to its native underling interfaces. When the target is a Server, it includes a first TLLAO in the NA message with Interface ID set to 255 and with its own link-layer address information, and also includes additional TLLAOs corresponding to the destination Client's Proxyed, NATed or VPNed underlying interfaces. (For NATed or VPNed underlying interfaces the server writes its own link-layer address in the TLLAO, and for Proxyed interfaces it writes the link-layer address of the Proxy.) When the source Proxy receives the NA message, it creates a dynamic neighbor cache entry in the FORWARD state that associates the TLLAOs of the NA message as the next-hop toward the routes advertised in the NA RIOs.

When a source Proxy sends route optimization NS messages toward the target, it can include RIOs to assert specific routes, and the target will only accept packets from the source Proxy with matching source addresses. If the source Proxy wishes to assert a "wildcard" route, it includes an RIO in the NS message with Prefix and Prefix Length set to 0. In that case, the target will either accept or ignore the NS based on its configured trust policy. If the target accepts the NS, it will accept all packets originating from the source Proxy regardless of their source address.

After the initial NS/NA exchange, the target may need to update the neighbor cache entries for any source Proxies for which it holds a

dynamic neighbor cache entry in the ACCEPT state. The target therefore sends unsolicited NA messages to announce any link layer changes. As a result:

- o the source Proxy may receive unsolicited NA messages with TLLAOs with new UDP Port Number, IP Address and/or QoS preferences from the target. In that case, the Proxy updates its neighbor cache entry and forwards future outbound packets based on the new link layer information.

- o the source Proxy may receive reflected packets destined to the link-layer address of a departed Client. In that case, the Proxy proceeds as discussed in Section 3.8.5.

- o the source Proxy may receive link-layer Destination Unreachable messages in response to data packets it sends to one of the target link-layer addresses. In that case, the Proxy processes the link-layer error messages as an indication that the path may be failing and proceeds as discussed in Section 3.13.

After the NS/NA exchange, while data packets are still flowing the source Proxy sends additional NS messages to the target using the address in the target's first TLLAO as the destination. The NS message will update the target's AcceptTime timer, and the resulting NA reply will update the source Proxy's ForwardTime timer in their respective neighbor cache entries.

If at some later time the target Client departs from its secured enclave, the Proxy sends unsolicited NAs to the Client's Servers to announce the departure.

5. Direct Underlying Interfaces

When a Client's AERO interface is configured over a direct underlying interface, the neighbor at the other end of the direct link can receive packets without any encapsulation. In that case, the Client sends packets over the direct link according to the QoS preferences associated with its underling interfaces. If the direct underlying interface has the highest QoS preference, then the Client's IP packets are transmitted directly to the peer without going through an underlying network. If other underlying interfaces have higher QoS preferences, then the Client's IP packets are transmitted via a different underlying interface, which may result in the inclusion of AERO Proxies, Servers and Relays in the communications path. Direct underlying interfaces must be tested periodically for reachability, e.g., via NUD, via periodic unsolicited NAs, etc.

6. Operation on AERO Links with /64 ASPs

IPv6 AERO links typically have ASPs that cover many candidate ACPs of length /64 or shorter. However, in some cases it may be desirable to use AERO over links that have only a /64 ASP. This can be accommodated by treating all Clients on the AERO link as simple hosts that receive /128 prefix delegations.

In that case, the Client sends an RS message to the Server the same as for ordinary AERO links. The Server responds with an RA message that includes one or more /128 prefixes (i.e., singleton addresses) that include the /64 ASP prefix along with an interface identifier portion to be assigned to the Client. The Client and Server then configure their AERO addresses based on the interface identifier portions of the /128s (i.e., the lower 64 bits) and not based on the /64 prefix (i.e., the upper 64 bits).

For example, if the ASP for the host-only IPv6 AERO link is 2001:db8:1000:2000::/64, each Client will receive one or more /128 IPv6 prefix delegations such as 2001:db8:1000:2000::1/128, 2001:db8:1000:2000::2/128, etc. When the Client receives the prefix delegations, it assigns the AERO addresses fe80::1, fe80::2, etc. to the AERO interface, and assigns the global IPv6 addresses (i.e., the /128s) to either the AERO interface or an internal virtual interface such as a loopback. In this arrangement, the Client conducts route optimization in the same sense as discussed in Section 3.15.

This specification has applicability for nodes that act as a Client on an "upstream" AERO link, but also act as a Server on "downstream" AERO links. More specifically, if the node acts as a Client to receive a /64 prefix from the upstream AERO link it can then act as a Server to provision /128s to Clients on downstream AERO links.

7. Implementation Status

An AERO implementation based on OpenVPN (https://openvpn.net/) was announced on the v6ops mailing list on January 10, 2018. The latest version is available at: http://linkupnetworks.net/aero/AERO-OpenVPN-1.0.tgz.

An initial public release of the AERO proof-of-concept source code was announced on the intarea mailing list on August 21, 2015. The latest version is available at: http://linkupnetworks.net/aero/aero-3.0.3a.tgz.

8. IANA Considerations

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

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

No further IANA actions are required.

9. 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 trusted Server.

NS and 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. NS and RS messages SHOULD include a Nonce option (see Section 5.3 of [RFC3971]) that recipients echo back in corresponding responses. In cases where spoofing cannot be mitigated through other means, however, all AERO IPv6 ND messages should employ SEND [RFC3971], which also protects the PD information embedded in RS/RA message options.

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, however AERO nodes SHOULD also use securing services such as SEND for Client authentication and network admission control. Following authenticated Client admission and prefix delegation procedures, AERO nodes MUST ensure that the source of data packets corresponds to the node to which the prefixes were delegated.

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

AERO Clients, Servers and Relays on the open Internet are susceptible to the same attack profiles as for any Internet nodes. For this reason, IP security SHOULD be used when AERO is employed over unmanaged/unsecured links using securing mechanisms such as IPsec [RFC4301], IKE [RFC5996] and/or TLS [RFC5246]. In some environments, however, the use of end-to-end security from Clients to correspondent nodes (i.e., other Clients and/or Internet nodes) could obviate the need for IP security between AERO Clients, Servers and Relays.

AERO Servers and Relays present targets for traffic amplification DoS attacks. This concern is no different than for widely-deployed VPN security gateways in the Internet, where attackers could send spoofed packets to the gateways at high data rates. This can be mitigated by connecting Relays and Servers over dedicated links with no connections to the Internet and/or when connections to the Internet are only permitted through well-managed firewalls.

Traffic amplification DoS attacks can also target an AERO Client's low data rate links. This is a concern not only for Clients located on the open Internet but also for Clients in secured enclaves. AERO Servers can institute rate limits that protect Clients from receiving packet floods that could DoS low data rate links.

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

10. Acknowledgements

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

This work has further been encouraged and supported by Boeing colleagues including Kyle Bae, M. Wayne Benson, Dave Bernhardt, Cam Brodie, Balaguruna Chidambaram, Irene Chin, Bruce Cornish, Claudiu Danilov, Wen Fang, Anthony Gregory, Jeff Holland, Ed King, Gene MacLean III, Rob Muszkiewicz, Sean O'Sullivan, Kent Shuey, Brian

Skeen, Mike Slane, Carrie Spiker, Brendan Williams, Julie Wulff, Yueli Yang, Eric Yeh and other members of the BR&T and BIT mobile networking teams. Wayne Benson, Kyle Bae and Eric Yeh are especially acknowledged for implementing the AERO functions as extensions to the public domain OpenVPN distribution.

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

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

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

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

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

- AERO, First Edition [RFC6706]

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

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

This work is aligned with the Boeing Information Technology (BIT) MobileNet program.

This work is aligned with the Boeing Research and Technology (BR&T) autonomous systems networking program.

11. References

11.1. Normative References

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

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

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

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

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

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

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

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

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

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

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

[RFC5175] Haberman, B., Ed. and R. Hinden, "IPv6 Router Advertisement Flags Option", RFC 5175, DOI 10.17487/RFC5175, March 2008, <https://www.rfc-editor.org/info/rfc5175>.

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

11.2. Informative References

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

[I-D.ietf-intarea-gue] Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-intarea-gue-05 (work in progress), December 2017.

[I-D.ietf-intarea-gue-extensions] Herbert, T., Yong, L., and F. Templin, "Extensions for Generic UDP Encapsulation", draft-ietf-intarea-gue-extensions-04 (work in progress), March 2018.

[I-D.ietf-intarea-tunnels] Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-08 (work in progress), January 2018.

[I-D.templin-6man-dhcpv6-ndopt] Templin, F., "IPv6 Neighbor Discovery Extensions for Prefix Delegation", draft-templin-6man-dhcpv6-ndopt-04 (work in progress), March 2018.

[I-D.templin-6man-rio-redirect] Templin, F. and j. woodyatt, "Route Information Options in IPv6 Neighbor Discovery", draft-templin-6man-rio-redirect-06 (work in progress), May 2018.

[I-D.templin-atn-bgp] Templin, F., Saccone, G., Dawra, G., and A. Lindem, "A Simple BGP-based Mobile Routing System for the Aeronautical Telecommunications Network", draft-templin-atn-bgp-06 (work in progress), March 2018.

[I-D.templin-intarea-grefrag]
Templin, F., "GRE Tunnel Level Fragmentation", draft-templin-intarea-grefrag-04 (work in progress), July 2016.

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

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

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

[I-D.templin-v6ops-pdhost]
Templin, F., "IPv6 Prefix Delegation Models", draft-templin-v6ops-pdhost-19 (work in progress), March 2018.

[OVPN] OpenVPN, O., "http://openvpn.net", October 2016.

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

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

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

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

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

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

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

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

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

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

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

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, DOI 10.17487/RFC2890, September 2000, <https://www.rfc-editor.org/info/rfc2890>.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Appendix A. AERO Alternate Encapsulations

When GUE encapsulation is not needed, AERO can use common encapsulations such as IP-in-IP [RFC2003][RFC2473][RFC4213], Generic Routing Encapsulation (GRE) [RFC2784][RFC2890] and others. The encapsulation is therefore only differentiated from non-AERO tunnels through the application of AERO control messaging and not through, e.g., a well-known UDP port number.

As for GUE encapsulation, alternate AERO encapsulation formats may require encapsulation layer fragmentation. For simple IP-in-IP encapsulation, an IPv6 fragment header is inserted directly between the inner and outer IP headers when needed, i.e., even if the outer header is IPv4. The IPv6 Fragment Header is identified to the outer IP layer by its IP protocol number, and the Next Header field in the IPv6 Fragment Header identifies the inner IP header version. For GRE encapsulation, a GRE fragment header is inserted within the GRE header [I-D.templin-intarea-grefrag].

Figure 5 shows the AERO IP-in-IP encapsulation format before any fragmentation is applied:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Outer IPv4 Header     |          |    Outer IPv6 Header      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|IPv6 Frag Header (optional)|          |IPv6 Frag Header (optional)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Inner IP Header      |          |       Inner IP Header     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           |          |                           |
~                           ~          ~                           ~
~    Inner Packet Body      ~          ~     Inner Packet Body     ~
~                           ~          ~                           ~
|                           |          |                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Minimal Encapsulation in IPv4      Minimal Encapsulation in IPv6
```

Figure 5: Minimal Encapsulation Format using IP-in-IP

Figure 6 shows the AERO GRE encapsulation format before any fragmentation is applied:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Outer IP Header        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          GRE Header           |
| (with checksum, key, etc..)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| GRE Fragment Header (optional)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Inner IP Header        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
~                               ~
~      Inner Packet Body        ~
~                               ~
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Minimal Encapsulation Using GRE

Alternate encapsulation may be preferred in environments where GUE encapsulation would add unnecessary overhead. For example, certain low-bandwidth wireless data links may benefit from a reduced encapsulation overhead.

GUE encapsulation can traverse network paths that are inaccessible to non-UDP encapsulations, e.g., for crossing Network Address Translators (NATs). More and more, network middleboxes are also being configured to discard packets that include anything other than a well-known IP protocol such as UDP and TCP. It may therefore be necessary to determine the potential for middlebox filtering before enabling alternate encapsulation in a given environment.

In addition to IP-in-IP, GRE and GUE, AERO can also use security encapsulations such as IPsec and SSL/TLS. In that case, AERO control messaging and route determination occur before security encapsulation is applied for outgoing packets and after security decapsulation is applied for incoming packets.

AERO is especially well suited for use with VPN system encapsulations such as OpenVPN [OVPN].

Appendix B. When to Insert an Encapsulation Fragment Header

An encapsulation fragment header is inserted when the AERO tunnel ingress needs to apply fragmentation to accommodate packets that must be delivered without loss due to a size restriction. Fragmentation is performed on the inner packet while encapsulating each inner packet fragment in outer IP and encapsulation layer headers that differ only in the fragment header fields.

The fragment header can also be inserted in order to include a coherent Identification value with each packet, e.g., to aid in Duplicate Packet Detection (DPD). In this way, network nodes can cache the Identification values of recently-seen packets and use the cached values to determine whether a newly-arrived packet is in fact a duplicate. The Identification value within each packet could further provide a rough indicator of packet reordering, e.g., in cases when the tunnel egress wishes to discard packets that are grossly out of order.

In some use cases, there may be operational assurance that no fragmentation of any kind will be necessary, or that only occasional large control messages will require fragmentation. In that case, the encapsulation fragment header can be omitted and ordinary fragmentation of the outer IP protocol version can be applied when necessary.

Appendix C. Autoconfiguration for Constrained Platforms

On some platforms (e.g., popular cell phone operating systems), the act of assigning a default IPv6 route and/or assigning an address to an interface may not be permitted from a user application due to

security policy. Typically, those platforms include a TUN/TAP interface [TUNTAP] that acts as a point-to-point conduit between user applications and the AERO interface. In that case, the Client can instead generate a "synthesized RA" message. The message conforms to [RFC4861] and is prepared as follows:

- o the IPv6 source address is the Client's AERO address
- o the IPv6 destination address is all-nodes multicast
- o the Router Lifetime is set to a time that is no longer than the ACP DHCPv6 lifetime
- o the message does not include a Source Link Layer Address Option (SLLAO)
- o the message includes a Prefix Information Option (PIO) with a /64 prefix taken from the ACP as the prefix for autoconfiguration

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

Appendix D. Operational Deployment Alternatives

AERO can be used in many different variations based on the specific use case. The following sections discuss variations that adhere to the AERO principles while allowing selective application of AERO components.

D.1. 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.

D.2. 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 PD authority through some means outside the scope of this document.

D.3. Operation on Client-less AERO Links

In some environments, the AERO service may be useful for mobile nodes that do not implement the AERO Client function and do not perform encapsulation. For example, if the mobile node has a way of injecting its ACP into the access subnetwork routing system an AERO Server connected to the same access network can accept the ACP prefix injection as an indication that a new mobile node has come onto the subnetwork. The Server can then inject the ACP into the BGP routing system the same as if an AERO Client/Server DHCPv6 PD exchange had occurred. If the mobile node subsequently withdraws the ACP from the access network routing system, the Server can then withdraw the ACP from the BGP routing system.

In this arrangement, AERO Servers and Relays are used in exactly the same ways as for environments where DHCPv6 Client/Server exchanges are supported. However, the access subnetwork routing systems must be capable of accommodating rapid ACP injections and withdrawals from mobile nodes with the understanding that the information must be propagated to all routers in the system. Operational experience has shown that this kind of routing system "churn" can lead to overall instability and routing system inconsistency.

D.4. 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-provisioned link-local address and exchange NS/NA messages the same as for dynamically-established tunnels.

D.5. Encapsulation Avoidance on Relay-Server Dedicated Links

In some environments, AERO Servers and Relays may be connected by dedicated point-to-point links, e.g., high speed fiberoptic leased lines. In that case, the Servers and Relays can participate in the AERO link the same as specified above but can avoid encapsulation over the dedicated links. In that case, however, the links would be dedicated for AERO and could not be multiplexed for both AERO and non-AERO communications.

D.6. 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 route optimization messages and continue to send packets via their Servers.

D.7. Extending AERO Links Through Security Gateways

When an enterprise mobile node 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 node to the security gateway. During this process, the mobile node supplies the security gateway with its public Internet address as the link-layer address for the VPN. The mobile node 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 node (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 nodes 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 nodes 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 node 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.

Appendix E. Change Log

Changes from -81 to -82:

- o Make DHCPv6 the default (but not exclusive) PD service
- o Support operation with no PD services nor ND Route Information Options
- o Updates to AERO Proxy function

Changes from -80 to -81:

- o Updates to Server and Proxy Extended Route Optimization
- o Updates to AERO Proxy section
- o Cleanups and clarifications

Changes from -79 to -80:

- o Substantial updates to AERO Proxy function
- o Removed 'V' bit from SLLAO and replaced with 'X' bit
- o Added concept of Direct, Proxyed, NATed, VPNed and Native underlying interfaces
- o Adjusted route optimization text according to underrlying interface types

Changes from -78 to -79:

- o Neighbors now set UDP Port Number and IP Address in S/TLLAOs to 0 if the node is behind a NAT or otherwise does not wish to update its link-layer address for this underlying interface
- o Introduced "proxy" as a new neighbor cache entry type
- o updated GUE references
- o multipath considerations for error message handling and NUD

Changes from -77 to -78:

- o Added "V" bit to SLLAO flags field for NS messages. V=1 indicates that the NA response must go through the reverse chain of Servers and Relays

- o Now including DHCPv6 PD messages as IPv6 ND message options

- o Clarified the use of the "P" bit in the RA flags field

- o Use of SEND to protect the combined DHCPv6/IPv6ND messages

- o Proxy now treats a Client's Servers as the default routers (i.e., instead of using a Relay as the default).

Changes from -76 to -77:

- o Now using IPv6 ND NS/NA messaging for route optimization (no longer using Predirect/Redirect)

- o Now using combined IPv6 ND/DHCPv6 messaging so autoconfiguration can be conducted in a single message exchange

- o Introduced the AERO Proxy construct. Critical for applications such as ATN/IPS

Changes from -75 to -76:

- o Bumped version number ahead of expiration deadline

Changes from -74 to -75:

- o Bumped version number ahead of expiration deadline

Author's Address

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

Email: fltemplin@acm.org

DMM Working Group
Internet Draft
Expires: July 2015

Zhiwei Yan
CNNIC
Jong-Hyouk Lee
Sangmyung University
Xiaodong Lee
CNNIC

January 9, 2015

Home Network Prefix Renumbering in PMIPv6
draft-yan-dmm-hnprenum-00.txt

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 will expire on July, 2015.

Copyright Notice

Table of Contents

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

[1] S. Jiang, B. Liu, and B. Carpenter, "IPv6 Enterprise Network Renumbering Scenarios and Guidelines", RFC 6879, February 2013.

[2] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil, "Proxy Mobile IPv6", RFC 5213, August 2008.

[3] J. Korhonen, S. Gundavelli, H. Yokota, and X. Cui, "Runtime Local Mobility Anchor (LMA) Assignment Support for Proxy Mobile IPv6", RFC 6463, February 2012.

[4] C. Perkins, D. Johnson, and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

[5] S. Krishnan, S. Gundavelli, M. Liebsch, H. Yokota and J. Korhonen, "Update Notifications for Proxy Mobile IPv6", RFC 7077, November 2013.

[6] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, November 2000.

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.