DMM Working Group
Internet-Draft
Intended status: Informational
Expires: March 12, 2018

CJ. Bernardos
A. de la Oliva
UC3M
F. Giust
NEC

September 8, 2017

An IPv6 Distributed Client Mobility Management approach using existing mechanisms
draft-bernardos-dmm-cmip-08

Abstract

The use of centralized mobility management approaches -- such as Mobile IPv6 -- poses some difficulties to operators of current and future networks, due to the expected large number of mobile users and their exigent demands. All this has triggered the need for distributed mobility management alternatives, that alleviate operators' concerns allowing for cheaper and more efficient network deployments.

This draft describes a possible way of achieving a distributed mobility behavior with Client Mobile IP, based on Mobile IPv6 and the use of Cryptographic Generated Addresses.

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 RFC 2119 [RFC2119].

Status of This Memo

Table of Contents

1. Introduction

Most of the currently standardized IP mobility solutions, like Mobile IPv6 [RFC6275], or Proxy Mobile IPv6 [RFC5213] rely to a certain extent on a centralized mobility anchor entity. This centralized network node is in charge of both the control of the network entities involved in the mobility management (i.e., it is a central point for the control signalling), and the user data forwarding (i.e., it is also a central point for the user plane). This makes centralized

mobility solutions prone to several problems and limitations, as identified in [RFC7333]: longer (sub-optimal) routing paths, scalability problems, signaling overhead (and most likely a longer associated handover latency), more complex network deployment, higher vulnerability due to the existence of a potential single point of failure, and lack of granularity on the mobility management service (i.e., mobility is offered on a per-node basis, not being possible to define finer granularity policies, as for example per-application).

There are basically two main approaches that are being researched now: one aimed at making Mobile IPv6 work in a distributed way, and another one doing the same exercise for Proxy Mobile IPv6 (see the document [RFC7429]). In this draft we describe a solution to achieve a DMM behavior with a CMIP (MIPv6) solution. This document is based on a research paper of the same authors, called "Flat Access and Mobility Architecture: an IPv6 Distributed Client Mobility Management solution" [GOB11].

2. Terminology

The following terms used in this document are defined in the Mobile IPv6 specification [RFC6275]:

Home Agent (HA)

Home Link

Home Address (HoA)

Care-of Address (CoA)

Binding Update (BU)

Binding Acknowledgement (BA)

The following terms are defined and used in this document:

DAR (Distributed Anchor Router). First hop routers where the mobile nodes attach to. They also play the role of mobility managers for the IPv6 addresses they anchor.

HDAR (Home Distributed Anchor Router). DAR which plays the role of Home Agent for a particular IPv6 address (i.e., DAR where that IPv6 address is anchored).

3. Description of the solution

Distributed Mobility Management approaches try to overcome the limitations of the traditional centralized mobility management, i.e., Mobile IP, by bringing the mobility anchor closer to the MN. Following this idea, in our approach -- that we call Flat Access and Mobility Architecture (FAMA) -- the MIPv6 centralized home agent is moved to the edge of the network, being deployed in the default gateway of the mobile node. That is, the first elements that provide IP connectivity to a set of MNs are also the mobility managers for those MNs. In the following we will call these access routers Distributed Anchor Routers (DARs).

The diagram in Figure 1 depicts the operations of the proposed solution. When a mobile node attaches to a distributed anchor router, it gets an IPv6 address which is topologically anchored at the DAR (Pref1::addr1 - HoA1). In the scheme we assume the address configuration takes place through a Router Solicitation/Router Advertisement handshake. While attached to this DAR, the mobile can send and receive traffic using HoA1 without traversing any tunneling nor special packet handling.

If the the mobile node moves to a different DAR, it gets a new IPv6 address from the new access router (Pref2::addr2 - HoA2). In case the MN wants to keep the reachability of the IPv6 address(es) it obtained from the previous DAR (note that this decision is dynamic and it is out of scope of this document, it can be done on an application basis for example), the host has to involve its MIPv6 stack, by sending a Binding Update to the DAR where the IPv6 address is anchored, using the address obtained from the current DAR as care-of address (in our example the MN binds HoA2 as CoA to DAR1).

```
    +-----+      +-----+      +-----+      +-----+      +-----+
    | MN  |      |DAR1 |      |DAR2 |      | CN1 |      | CN2 |
    +-----+      +-----+      +-----+      +-----+      +-----+
       |            |            |            |            |
       |-Rtr. Sol.->|            |            |            |
       |<-Rtr. Adv.-|            |            |            |
       |            |            |            |            |
   Pref1::addr1     |            |            |            |
   conf. (HoA1)     |            |            |            |
       |            |            |            |            |
       |<-----------+--IP flow 1------------->|            |
       |            |            |            |            |
   - o - o - o - o - o Handover to DAR2 o - o - o - o - o - o -
       |            |            |            |            |
       |--------Rtr. Sol.------->|            |            |
       |<-------Rtr. Adv.--------|            |            |
   Pref2::addr2     |            |            |            |
   conf. (HoA2)     |            |            |            |
       |----BU----->|            |            |            |
       | (CoA=HoA2) |            |            |            |
       |<---BA------|            |            |            |
       |(-tnl est.-)|            |            |            |
       |            |            |            |            |
       |(<---------)+--IP flow 1------------->|            |
       |            |            |            |            |
       |<--------------IP flow 2-+------------------------>|
       |            |            |            |            |
```

Figure 1: Signalling after the first handover

In this way, the IPv6 address that the node wants to maintain in use (Pref1::addr1) plays the role of home address (HoA1), and the DAR from where that address was configured plays the role of Home Agent (for that particular address). In this scenario, old flows are anchored to the previous DAR (DAR1), which is in charge to encaspulate the packets and deliver them to the MN’s CoA. The IP tunnel is bi-directional, so the MN does the same when sending packets with the old address (HoA1). Conversely, new IP flows are started using the address configured at the new DAR (HoA2). These flows are handled by the new DAR as a plain IPv6 router.

Note that the FAMA approach basically enables a mobile node to simultaneously handle several IPv6 addresses -- each of them anchored at a different DAR -- ensuring their continuous reachability by using Mobile IPv6 in a distributed fashion (i.e., each access router is a potential home agent for the address it delegates, if required). Figure 2 illustrates the above case in which the MN is connected to DAR2, but flow1 is anchored at DAR1, because it was started by the MN

using the IPv6 address Pref1::addr1, configured when the MN was connected to DAR1. In the same example, the MN starts flow2 using Pref2::addr2, assigned by DAR2.

```
                 +---+     +---+
                 |CN1|     |CN2|
                 +*--+     +*--+
                  *         *
                  *         *
               *****        *
        flow1 *             * flow2
              *             *
              *             *
              *             *
          +---*-+      +----*+     +-----+
          |   * | ___  |    *|     |     |
          |DAR1**(__*\-+DAR2*+-----+DAR3 |
          |     |   \*\|    *|     |     |
          +-----+    \*\----*+     +-----+
                      \*\   *
                       \*\  *
            flow1 using \*--*+ flow2 using
           Pref1::addr1 |*  *| Pref2::addr2
                        |*MN*|
                        +----+

  Operations sequence                  Packets flow
```

Figure 2: MN’s flows forwarding in FAMA

The same operations take place if the MN moves to another DAR. The MN obtains a new address (Pref3::addr3 - HoA3), which is indicated as CoA in the BU messages sent by the MN to the previous DARs. This distributed address anchoring is enabled on demand and on a per-address granularity, which means that depending on the user needs, it might be the case that all, some or none of the IPv6 addresses that a mobile node configures while moving within a FAMA domain, are kept reachable and used by the mobile. The scheme in Figure 3 depicts the example where the MN updates all the previous DARs, mapping the corresponding HoA with the new CoA

```
    +-----+      +-----+      +-----+      +-----+      +-----+
    | MN  |      |DAR1 |      |DAR2 |      |DAR3 |      | CNs |
    +-----+      +-----+      +-----+      +-----+      +-----+
       |            |            |            |            |
       (<-----------)--IP flow 1-------------------------->|
       |            |            |            |            |
       |<--------------IP flow 2-+------------------------>|
       |            |            |            |            |
   - o - o - o - o - o Handover to DAR2 o - o - o - o - o - o -
       |            |            |            |            |
       |--------------Rtr. Sol.-------------->|            |
       |<-------------Rtr. Adv.---------------|            |
   Pref3::addr3     |            |            |            |
   conf. (HoA3)     |            |            |            |
       |----BU----->|            |            |            |
       | (CoA=HoA)  |            |            |            |
       |<---BA------|            |            |            |
       |(-tnl est.-)|            |            |            |
       |------------BU---------->|            |            |
       |        (CoA=HoA3)       |            |            |
       |<-----------BA-----------|            |            |
       |(---------tnl est.------)|            |            |
       |            |            |            |            |
       |(<---------)+-IP flow 1--------------------------->|
       |            |            |            |            |
       |(<------------IP flow 2-)+------------------------>|
       |            |            |            |            |
       |<-------------IP flow 3---------------+----------->|
       |            |            |            |            |
```

Figure 3: Signalling after a second handover

In traditional Mobile IPv6, the communication between the MN and the HA is secured through IPsec [RFC4877]. Following a similar approach in FAMA is difficult due to the large number of security associations that would be required, since any gateway of the access network can play the role of home agent for any mobile node. In order to overcome this problem and provide authentication between the DAR and the MNs, we propose the use of Cryptographically Generated Addresses [RFC3972] (CGAs), as introduced in [I-D.laganier-mext-cga]. CGAs are a powerful mechanism allowing authentication of the packets and requires no public-key infrastructure, hence it is well-suited for this application.

Following the ideas presented above, every time an MN attaches to a DAR, it configures a CGA from a prefix anchored at the DAR (e.g., by using stateless address auto-configuration mechanisms). This address can then be used by the MN to establish a communication with a remote

Correspondent Node (CN) while attached to that particular DAR. If the mobile then moves to a new DAR (nDAR), the following two cases are possible: i) there is no need for the address that was configured at the previous DAR (pDAR) to survive the movement: in this case there is no further action required; ii) the mobile wants to keep the reachability of the address configured at pDAR: in this case Mobile IPv6 is triggered, and the MN sends a Binding Update (BU) message to the pDAR, using the address configured at the previous DAR as home address, and the address configured at the new DAR as care-of address. This BU includes the CGA parameters and signature [I-D.laganier-mext-cga], which are used by the receiving DAR to identify the MN as the legitimate owner of the address. Although the use of CGAs does not impose a heavy burden in terms of performance, depending on the number of MNs handled at the DAR, the processing of the CGAs can be problematic. To reduce the complexity of the proposed protocol, we suggest an alternative mechanism to authenticate any subsequent signaling packets exchanged between the MN and the DAR (in case the mobile performs a new attachment to a different DAR). This alternative method relies on the use of a Permanent Home Keygen Token (PHKT), which will be used to generate the Authorization option that the MN has to include in all next Binding Update messages. This token is forwarded to the MN in the Binding Acknowledgment message, sent on reply to the BU. The procedure is depicted in Figure 4. Once the signaling procedure is completed, a bi-directional tunnel is established between the mobile node and the DAR where the IPv6 address is anchored (the "home" DAR -- HDAR -- for that particular address), so the mobile can continue using the IPv6 address.

```
           ------                                 -------
           | MN |                                 | DAR |
           ------                                 -------
             |                                       |
        CGA  |                                       |
     config  |-- BU + CGA param + signature ------>|
             |                                       |  MN
      PHKT   |<------------------------ BA + PHKT --| auth
    caching  |                                       |
             |                                       |
                        (first handoff)

     PHKT    |                                       |
    refresh, |                                       |
     next    |-- BU(PHKT auth) ------------------->|
   handoffs, |                                       |  MN
     de-reg  |<------------------------------- BA --| auth
             |                                       |
             |                                       |
                     (subsequent signaling)
```

Figure 4: Signaling between the MN and the DAR

In case the MN performs any subsequent movements and it requires to maintain the reachability of an address for which it has already sent a BU, the following BU messages can be secured using the PHKT exchanged before, reducing the computational load at the receiving DAR.

Note that on every attachment of a node to a DAR, the terminal also obtains a new IPv6 address which is topologically anchored at that DAR, and that this address can be used for new communications (avoiding in this way the tunneling required when using an address anchored at a different DAR). A mobile can keep multiple IPv6 addresses active and reachable at a given time, and that requires to send -- every time the MN moves -- a BU message to all the previous DARs that are anchoring the IP flows that the MN wish to maintain.

4. IANA Considerations

TBD.

5. Security Considerations

Although the approach documented in this document is attractive for the reduced signaling overhead caused by the mobility support, it can be misused in some particular scenarios by malicious nodes that wish to export an incorrect CoA in the BU message, since it does provide

proof of the MN's reachability at the visited network. Indeed, the CGA approach assures that the BU message has been sent by the legitimate HoA's owner but it does not make sure that same MN to be reachable at the CoA indicated. This requires further analysis.

A possible approach to provide a more secure solution is the following: a Return Routability procedure similar to the one defined in MIPv6 Route Optimization can be used to mitigate the aforementioned security issue. The Return Routability procedure starts after the handoff. Instead of sending the BU message, the MN sends a Care-of Test Init message (CoTI). This message is replied by the DAR with a Care-Of Test message containing a CoA Keygen Token. The MN can now send a BU using both Home and CoA Keygen tokens to proof its reachability at both the HoA and the CoA. The message and the knowledge of both tokens is a proof that the MN is the legitimate node who has sent the BU and also is reachable at the CoA indicated. As all security improvements, the one proposed incurs in a performance penalty, in this case an increase in the handover delay. Specifically this enhanced security approach requires four messages to be exchanged between the MN and the DAR instead of the two messages of the original solution. In terms of handover delay, it increases it by a factor of two, as the new solution requires to two Round Trip Times (RTTs) to conclude, instead of one.

6. References

6.1. Normative References

[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>.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <https://www.rfc-editor.org/info/rfc3972>.

[RFC4877] Devarapalli, V. and F. Dupont, "Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture", RFC 4877, DOI 10.17487/RFC4877, April 2007, <https://www.rfc-editor.org/info/rfc4877>.

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

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

6.2. Informative References

[GOB11] Giust, F., de la Oliva, A., and CJ. Bernardos, "Flat Access and Mobility Architecture: an IPv6 Distributed Client Mobility Management solution", 3rd IEEE International Workshop on Mobility Management in the Networks of the Future World (MobiWorld 2011), colocated with IEEE INFOCOM 2011 , 2011.

[I-D.laganier-mext-cga] Laganier, J., "Authorizing Mobile IPv6 Binding Update with Cryptographically Generated Addresses", draft-laganier-mext-cga-01 (work in progress), October 2010.

[RFC7333] Chan, H., Ed., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", RFC 7333, DOI 10.17487/RFC7333, August 2014, <https://www.rfc-editor.org/info/rfc7333>.

[RFC7429] Liu, D., Ed., Zuniga, JC., Ed., Seite, P., Chan, H., and CJ. Bernardos, "Distributed Mobility Management: Current Practices and Gap Analysis", RFC 7429, DOI 10.17487/RFC7429, January 2015, <https://www.rfc-editor.org/info/rfc7429>.

Appendix A. Comparison with Requirement document

In this section we descrbe how our solution addresses the DMM requirements listed in [RFC7333].

A.1. Distributed mobility management

"IP mobility, network access solutions, and forwarding solutions provided by DMM MUST enable traffic to avoid traversing a single mobility anchor far from the optimal route."

In our solution, a DAR is responsible to handle the mobility for those IP flows started when the MN is attached to it. As long as the MN remains connected to the DAR’s access links, the IP packets of such flows can benefit from the optimal path. When the MN moves to another DAR, the path becomes non-optimal for ongoing flows, as they are anchored to the previous DAR, but newly started IP sessions are forwarded by the new DAR through the optimal path.

A.2. Bypassable network-layer mobility support for each application session

"DMM solutions MUST enable network-layer mobility, but it MUST be possible for any individual active application session (flow) to not use it. Mobility support is needed, for example, when a mobile host moves and an application cannot cope with a change in the IP address. Mobility support is also needed when a mobile router changes its IP address as it moves together with a host and, in the presence of ingress filtering, an application in the host is interrupted. However, mobility support at the network layer is not always needed; a mobile node can often be stationary, and mobility support can also be provided at other layers. It is then not always necessary to maintain a stable IP address or prefix for an active application session."

Our DMM solution operates at the IP layer, hence upper layers are totally transparent to the mobility operations. In particular, ongoing IP sessions are not disrupted after a change of access network. The routability of the old address is ensured by the IP tunnel with the old DAR. New IP sessions are started with the new address. From the application’s perspective, those processes which sockets are bound to a unique IP address do not suffer any impact. For the other applications, the sockets bound to the old address are preserved, whereas next sockets use the new address.

A.3. IPv6 deployment

"DMM solutions SHOULD target IPv6 as the primary deployment environment and SHOULD NOT be tailored specifically to support IPv4, particularly in situations where private IPv4 addresses and/or NATs are used."

The DMM solution we propose targets IPv6 only.

A.4. Existing mobility protocols

"A DMM solution MUST first consider reusing and extending IETF standard protocols before specifying new protocols."

This DMM solution is derived from the operations and messages specified in [RFC6275], [RFC3972], and [I-D.laganier-mext-cga].

A.5. Coexistence with deployed networks/hosts and operability across different networks

"A DMM solution may require loose, tight, or no integration into existing mobility protocols and host IP stacks. Regardless of the integration level, DMM implementations MUST be able to coexist with existing network deployments, end hosts, and routers that may or may not implement existing mobility protocols. Furthermore, a DMM solution SHOULD work across different networks, possibly operated as separate administrative domains, when the needed mobility management signaling, forwarding, and network access are allowed by the trust relationship between them"

The proposed solution can provide a fallback mechanism employing legacy Mobile IPv6, for instance forcing the MN to use only one DAR. Moreover, this solution applies when the MN is connected to an administrative domain not supporting trust relationships. Indeed, all the IP sessions can remain anchored to the DARs of the "home" domain. Our solution can be deployed across different domains with trust agreements.

A.6. Operation and management considerations

"A DMM solution needs to consider configuring a device, monitoring the current operational state of a device, and responding to events that impact the device, possibly by modifying the configuration and storing the data in a format that can be analyzed later.

The proposed solution can re-use existing mechanisms defined for the operation and management of Mobile IPv6.

A.7. Security considerations

"A DMM solution MUST support any security protocols and mechanisms needed to secure the network and to make continuous security improvements. In addition, with security taken into consideration early in the design, a DMM solution MUST NOT introduce new security risks or amplify existing security risks that cannot be mitigated by existing security protocols and mechanisms."

The proposed solution uses a CGA-based security system to enable authentication and authorization of mobile hosts.

A.8. Multicast

"DMM SHOULD enable multicast solutions to be developed to avoid network inefficiency in multicast traffic delivery."

This solution does not include multicast traffic in its scope. Nevertheless, it allows combining multicast support solutions, such as local subscription at each DAR, which would result in a flexible distribution scenario.

Appendix B. Open DMM platform

The client-based DMM solution described in this document is available at the Open Distributed Mobility Management (ODMM) project (http://www.odmm.net/). The ODMM platform is intended to foster DMM development and deployment, by serving as a framework to host open source implementations.

Authors' Addresses

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Antonio de la Oliva
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 8803
Email: aoliva@it.uc3m.es
URI: http://www.it.uc3m.es/aoliva/

Fabio Giust
NEC Laboratories Europe
NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg D-69115
Germany

Phone: +49 6221 4342216
Email: fabio.giust@neclab.eu

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 30, 2017

CJ. Bernardos
UC3M
JC. Zuniga
SIGFOX

May 29, 2017

PMIPv6-based distributed anchoring
draft-bernardos-dmm-distributed-anchoring-09

Abstract

Distributed Mobility Management solutions allow for setting up networks so that traffic is distributed in an optimal way and does not rely on centralized deployed anchors to provide IP mobility support.

There are many different approaches to address Distributed Mobility Management, as for example extending network-based mobility protocols (like Proxy Mobile IPv6), or client-based mobility protocols (as Mobile IPv6), among others. This document follows the former approach, and proposes a solution based on Proxy Mobile IPv6 in which mobility sessions are anchored at the last IP hop router (called distributed gateway). The distributed gateway is an enhanced access router which is also able to operate as local mobility anchor or mobility access gateway, on a per prefix basis. The draft focuses on the required extensions to effectively support simultaneously anchoring several flows at different distributed gateways.

This draft introduces the concept of distributed logical interface (at the distributed gateway), which is a software construct that allows to easily hide the change of anchor from the mobile node. Additionally, the draft describes how to provide session continuity in inter-domain scenarios in which dynamic tunneling or signaling between distributed gateways from different operators is not allowed.

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 RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Distributed Mobility Management (DMM) paradigm aims at minimizing the impact of currently standardized mobility management solutions, which are centralized (at least to a considerable extent).

Centralized mobility solutions, such as Mobile IPv6 or the different macro-level mobility management solutions of 3GPP EPS, base their operation on the existence of a central entity (e.g., HA, LMA, PGW or GGSN) that anchors the IP address used by the mobile node and is in charge of coordinating the mobility management (MM) (sometimes helped by a third entity like the MME or the HSS). This central anchor point is in charge of tracking the location of the mobile and redirecting its traffic towards its current topological location. While this way of addressing mobility management has been fully developed by the Mobile IP protocol family and its many extensions, there are also several limitations that have been identified [RFC7333]. Among them, we can just highlight sub-optimal routing, scalability problems (in the network and in the centralized anchor) and reliability [RFC7333].

Several DMM-based approaches are being proposed and explored now [RFC7429], [commag.dmm-standards]. One of them is based on extending network-based mobility protocols (such as Proxy Mobile IPv6 [RFC5213] or GTP) to operate in distributed fashion. This document proposes a solution that falls in this category, defining a new logical entity, called Distributed Gateway (D-GW) which basically encompasses the functionalities of plain IPv6 access router, MAG and LMA, on a per-IPv6 prefix basis. The main contribution of this draft is the definition of the mechanisms required to support the operation of such a network-based mobility solution when several flows are simultaneously anchored [I-D.ietf-dmm-distributed-mobility-anchoring] at different D-GWs, by introducing the concept of Distributed Logical

Interface (DLIF). The document also defines the required PMIPv6 signaling extensions. Last, but not least, the solution is also extended to provide session continuity across different domains.

2. Terminology

The following terms used in this document are defined in the Proxy Mobile IPv6 specification [RFC5213]:

Local Mobility Anchor (LMA)

Mobile Access Gateway (MAG)

Mobile Node (MN)

Binding Cache Entry (BCE)

Proxy Care-of Address (P-CoA)

Proxy Binding Update (PBU)

Proxy Binding Acknowledgment (PBA)

The following terms are defined and/or used in this document:

D-GW (Distributed Gateway). First IP hop router used by the mobile node. It provides an IPv6 prefix (topologically anchored at the D-GW) to each attaching mobile node.

Anchoring D-GW. A previously visited D-GW anchoring an IPv6 prefix which is still used by a mobile node.

Serving D-GW. The D-GW the MN is currently attached to.

DLIF (Distributed Logical Interface). It is a logical interface at the IP stack of the D-GW. For each active prefix used by the mobile node, the serving D-GW has a DLIF configured (associated to the anchoring D-GW). In this way, a serving D-GW exposes itself towards each MN as multiple routers, one per active anchoring D-GW.

HSS (Home Subscriber Server). In a 3GPP architecture, it is the master user database that contains the subscription-related information (subscriber profiles), performs authentication and authorization of the user, and can provide information about the subscriber’s location and IP information.

3. Solution’s overview

A new logical network entity, called Distributed Gateway (D-GW) is introduced at the edge of the network, close to the MN. It implements the functionality of a plain IPv6 access router (AR), a mobile access gateway (MAG) and a local mobility anchor (LMA), on a per-MN and per-IPv6-prefix, as described later.

The solution basically extends Proxy Mobile IPv6 [RFC5213] to behave in a distributed fashion, similarly as what has been proposed in [I-D.seite-dmm-dma] and [I-D.bernardos-dmm-pmip]. This is achieved by the D-GW logically behaving as a distributed mobility anchor, which comprises the following:

- o When a mobile node attaches to a D-GW (initial attachment or handover), the D-GW provides an IPv6 prefix to the MN, acting as a regular IPv6 router (with the only difference that the delegated prefix is only assigned to one single MN, not being shared with any other node). The D-GW that the mobile node is currently attached to is called "serving D-GW".

- o When a mobile node performs a handover, it attaches to a new D-GW and configures a new IPv6 address out of the prefix provided and anchored by the new serving D-GW. As before, the serving D-GW behaves as a plain IPv6 router for that particular MN and the delegated (locally anchored) prefix. If the MN has active traffic using addresses anchored by other D-GWs (which are called "anchoring D-GWs") or it just needs to keep the reachability of these addresses, the current serving D-GW also acts as MAG, by sending the required proxy binding update (PBU) to the corresponding anchoring D-GWs. The anchoring D-GWs therefore behave as LMA for this particular MN and the IPv6 prefixes they are anchoring, replying with a PBA.

- o Once the PBU/PBA signaling is completed, a bidirectional tunnel is established between the serving D-GW and the anchoring D-GW (one per D-GW anchoring an active prefix used by the MN). These tunnels are used to provide IP address continuity to prefixes that are not anchored at the serving D-GW.

- o The means for a serving D-GW to obtain the information about the prefixes that a locally attached mobile node wants to keep reachable, and the associated anchoring D-GWs are out of the scope of this draft. Among the possible mechanisms that can be used to let the D-GW know about the prefixes that should be kept reachable, we can cite for instance layer-2 triggers/signaling. Regarding the mapping of IPv6 prefixes to anchoring D-GWs, there might be either fully distributed mechanisms in place, or the

information can be maintained in a centralized repository (e.g., in the HSS, using a centralized LMA [I-D.bernardos-dmm-pmip], etc.).

The basic operation of the solution is shown with an example in Figure 1. MN1 attaches to D-GW1 (thus becoming its serving D-GW) and configures an IPv6 address (prefA::MN1) out of a prefix locally anchored at D-GW1 (prefA::/64). At this point, MN1 can communicate with any correspondent node of the Internet, being the traffic anchored at D-GW1. If later on MN1 moves to D-GW2, a new IPv6 address (PrefB::MN1) is configured by the mobile node, this time out of prefB::/64, which is anchored at D-GW2 (which becomes the new serving D-GW). D-GW2 also exchanges the required PBU/PBA signaling to ensure that data traffic using prefA::MN1 still reaches the mobile node, by setting up a bidirectional tunnel between D-GW1 (anchoring D-GW) and D-GW2 (serving D-GW).

```
           +-----+      +-------+      +-------+      +-------------+
           | MN1 |      | D-GW1 |      | D-GW2 |      | CN@Internet |
           +-----+      +-------+      +-------+      +-------------+
              |             |              |                 |
              |  attachment |              |                 |
              |<...........>|              |                 |
              |  prefA::/64 |              |                 |
              |<------------|              |                 |
   configures |             |              |                 |
   prefA::MN1 |             |              |                 |
              |          (traffic using prefA::MN1)          |
              |<------------|------------------------------>|
              |             |              |                 |
              |         handover           |                 |
             /............................./                 |
              |             |   prefB::/64 |                 |
              |<----------------------------|                 |
   configures |             |         PBU  |                 |
   prefB::MN1 |      tunnel |<-------------|                 |
   and keeps  |      set-up |  PBA         |                 |
     using    |             |------------->| tunnel          |
   prefA::MN1 |             |              | set-up          |
              |          (traffic using prefB::MN1)          |
              |<----------------------------|---------------->|
              |          (traffic using prefA::MN1)          |
              |             |<------------------------------>|
              |             |<============>|                 |
              |<--------------------------->|                 |
              |             |              |                 |
```

Figure 1: Basic operation of the solution

The next sections of this draft focus on the detailed operation of the D-GWs when a mobile node has multiple flows anchored at different distributed gateways.

4. Simultaneous anchoring of multiple flows (single operator)

In this section we describe the mechanisms required in the network to enable simultaneous anchoring of several flows at different D-GWs within the same operator.

4.1. The Distributed Logical Interface (DLIF) concept

One of the main challenges of a network-based DMM solution is how to allow a mobile node to simultaneously send/receive traffic which is anchored at different D-GWs, and how to influence on the preference of the mobile selecting the source IPv6 address for a new communication, without requiring special support on the mobile node stack. This document defines the Distributed Logical Interface (DLIF), which is a software construct that allows to easily hide the change of anchor from the mobile node.

```
  +-----------------------------------------------------+
 (                      Operator’s                      )
 (                         core                         )
  +-----------------------------------------------------+
           |                                 |
    +---------------+     tunnel    +---------------+
    |   IP  stack   |===============|   IP  stack   |
    +---------------+               +-------+-------+
    |    mn1dgw1    |--+ (DLIFs) +--|mn1dgw1|mn1dgw2|--+
    +---------------+  |         |  +-------+-------+  |
    | phy interface |  |         |  | phy interface |  |
    +---------------+  |         |  +---------------+  |
          D-GW1       (o)       (o)       D-GW2       (o)
                                   x                 x
                                     x             x
                        prefA::/64     x         x   prefB::/64
                      (AdvPrefLft=0)     x     x
                                           (o)
                                            |
                                         +-----+
                              prefA::MN1  | MN1 |  prefB::MN1
                             (deprecated) +-----+
```

Figure 2: DLIF: exposing multiple routers (one per active anchoring D-GW)

The basic idea of the DLIF concept is the following. Each serving D-GW exposes itself towards a given MN as multiple routers, one per active anchoring D-GW associated to the MN. Let’s consider the example shown in Figure 2, MN1 initially attaches to D-GW1, configuring an IPv6 address (prefA::MN1) from a prefix locally anchored at D-GW1 (prefA::/64). At this stage, D-GW1 plays both the role of anchoring and serving D-GW, and also it behaves as a plain IPv6 access router. D-GW1 creates a distributed logical interface to communicate (point-to-point link) with MN1, exposing itself as a (logical) router with a specific MAC (e.g., 00:11:22:33:01:01) and IPv6 addresses (e.g., prefA::DGW1/64 and fe80:211:22ff:fe33:101/64) using the DLIF mn1dgw1. As explained below, these addresses represent the "logical" identity of D-GW1 towards MN1, and will "follow" the mobile node while roaming within the domain (note that the place where all this information is maintained and updated is out-of-scope of this draft; potential examples are to keep it on the HSS or the user’s profile).

If MN1 moves and attaches to a different D-GW of the domain (D-GW2 in the example of Figure 2), this D-GW will create a new logical interface (mn1dgw2) to expose itself towards MN1, providing it with a locally anchored prefix (prefB::/64). In this case, since the MN1 has another active IPv6 address anchored at a D-GW1, D-GW2 also needs to create an additional logical interface configured to exactly resemble the one used by D-GW1 to communicate with MN1. In this example, there is only one active anchoring D-GW (in addition to D-GW2, which is the serving one): D-GW1, so only the logical interface mn1dgw1 is created, but the same process would be repeated in case there were more active anchoring D-GWs involved. In order to maintain the prefix anchored at D-GW1 reachable, a tunnel between D-GW1 and D-GW2 is established and the routing is modified accordingly. The PBU/PBA signaling is used to set-up the bi-directional tunnel between D-GW1 and D-GW2, and it might also be used to convey to D-GW2 the information about the prefix(es) anchored at D-GW1 and about the addresses of the associated DLIF (i.e., mn1dgw1).

```
   +-----------------------------------------+ +---------------------+
   |                   D-GW1                 | |         D-GW2       |
   |+----------------------------------------+| |+-------------------+|
   ||+-----------------++-----------------+|| ||+-----------------+||
   |||+-------++-------+||+-------++-------+||| |||+-------++-------+|||
   ||||mn3dgw1||mn3dgw2||||mn2dgw1||mn2dgw2|||| ||||mn1dgw1||mn1dgw2||||
   |||| LMAC1 || LMAC2 |||| LMAC3 || LMAC4 |||| |||| LMAC5 || LMAC6 ||||
   |||+-------++-------+||+-------++-------+||| |||+-------++-------+|||
   |||   LIFs of MN3    ||   LIFs of MN2    ||| |||   LIFs of MN1    |||
   ||+-----------------++-----------------+|| ||+-----------------+||
   ||                 HMAC1   (phy if D-GW1) || ||HMAC2 (phy if D-GW2)||
   |+----------------------------------------+| |+-------------------+|
   +-----------------------------------------+ +---------------------+
                     x        x                           x
                    x         x                          x
                  (o)         (o)                      (o)
                   |           |                        |
                +--+--+     +--+--+                  +--+--+
                | MN3 |     | MN2 |                  | MN1 |
                +-----+     +-----+                  +-----+
```

Figure 3: Distributed Logical Interface concept

Figure 3 shows the logical interface concept in more detail. The figure shows two D-GWs and three MNs. D-GW1 is currently serving MN2 and MN3, while D-GW2 is serving MN1. MN1, MN2 and MN3 have two active anchoring D-GWs: D-GW1 and D-GW2. Note that a serving D-GW always plays the role of anchoring D-GW for the attached (served) MNs. Each D-GW has one single physical wireless interface.

As introduced before, each MN always "sees" multiple logical routers -- one per active anchoring D-GW -- independently of to which serving D-GW the MN is currently attached. From the point of view of the MN, these D-GWs are portrayed as different routers, although the MN is physically attached to one single interface . The way this is achieved is by the serving D-GW configuring different logical interfaces. If we focus on MN1, it is currently attached to D-GW2 (i.e., D-GW2 is its serving D-GW) and, therefore, it has configured an IPv6 address from D-GW2’s pool (e.g., prefB::/64). D-GW2 has set-up a logical interface (mn1dgw2) on top of its wireless physical interface (phy if D-GW2) which is used to serve MN1. This interface has a logical MAC address (LMAC6), different from the hardware MAC address (HMAC2) of the physical interface of D-GW2. Over the mn1dgw2 interface, D-GW2 advertises its locally anchored prefix prefB::/64. Before attaching to D-GW2, MN1 visited D-GW1, configuring also an address locally anchored at this D-GW, which is still being used by the MN1 in active communications. MN1 keeps "seeing" an interface connecting to D-GW1, as if it were directly connected to the two

D-GWs. This is achieved by the serving D-GW (D-GW1) configuring an additional distributed logical interface: mn1dgw1, which behaves exactly as the logical interface configured by the actual D-GW1 when MN1 was attached to it. This means that both the MAC and IPv6 addresses configured on this logical interface remain the same regardless of the physical D-GW which is serving the MN. The information required by a serving D-GW to properly configure this logical interfaces can be obtained in different ways: as part of the information conveyed in the PBA, from an external database (e.g., the HSS) or by other means. As shown in the figure, each D-GW may have several logical interfaces associated to each attached MN, having always at least one (since a serving D-GW is also an anchoring D-GW for the attached MN).

In order to enforce the use of the prefix locally anchored at the serving D-GW, the router advertisements sent over those logical interfaces playing the role of anchoring D-GWs (different from the serving one) include a zero prefix lifetime. The goal is to deprecate the prefixes delegated by these D-GWs (which will be no longer serving the MN). Note that on-going communications keep on using those addresses, even if they are deprecated, so this only affects to new sessions.

The distributed logical interface concept also enables the following use case. Suppose that access to a local IP network is provided by a given D-GW (e.g., D-GW1 in the example shown in Figure 2) and that the resources available at that network cannot be reached from outside the local network (e.g., cannot be accessed by an MN attached to D-GW2). This is similar to the LIPA scenario currently being consider by 3GPP. The goal is to allow an MN to be able to roam while still being able to have connectivity to this local IP network. The solution adopted to support this case makes use of RFC 4191 [RFC4191] more specific routes when the MN moves to a D-GW different from the one providing access to the local IP network (D-GW1 in the example). These routes are advertised through the distributed logical interface representing the D-GW providing access to the local network (D-GW1 in this example). In this way, if MN1 moves from D-GW1 to D-GW2, any active session that MN1 may have with a node of the local network connected to D-GW1 will survive, being the traffic forwarded via the tunnel between D-GW1 and D-GW2. Also, any potential future connection attempt towards the local network will be supported, even though MN1 is no longer attached to D-GW1.

4.2. D-GW protocol operation

This section describes the D-GW operation in more detail.

Figure 4 shows an example of the D-GW operation:

1. MN1 attaches to D-GW1. This event is detected by D-GW1 (based on layer 2 signaling/triggers or the reception of a Router Solicitation sent by MN1).

2. An IPv6 prefix from the pool of locally anchored prefixes is selected by D-GW1 to be delegated to MN1 (prefA::/64). D-GW1 sets up a distributed logical interface aimed at interfacing with MN1, called mn1dgw1. D-GW1 starts sending router advertisements to MN1, including the delegated prefix.

3. D-GW1 learns if it is an attachment due to a handover (how this is done is out-of-scope of this draft). In this case it is an initial attachment, so nothing else is required.

4. The DLIF mn1dgw1 is used by D-GW1 to advertise the locally anchored prefix (prefA::/64) to MN1. Using this prefix, MN1 configures an IPv6 address (prefA::MN1/64) that can be used to start new sessions (which will be anchored at D-GW1). Traffic using the address prefA::MN1 is received at the interface mn1dgw1 and directly forwarded by D-GW1 towards its destination. Traffic between MN1 and the local network reachable via D-GW1 (localnet) is handled normally by D-GW1 (as MN1 is locally attached).

5. MN1 performs a handover to D-GW2. This event is detected by D-GW2.

6. An IPv6 prefix from the pool of locally anchored prefixes is selected by D-GW2 to be delegated to MN1 (prefB::/64). D-GW2 sets up a distributed logical interface aimed at interfacing with MN1, called mn1dgw2. D-GW2 starts sending router advertisements to MN1, including the delegated prefix. Traffic using the address prefB::MN1 is received at the interface mn1dgw2 and directly forwarded by D-GW2 towards its destination.

7. D-GW2 learns that this is a handover of MN1, and that it previously visited D-GW1. D-GW2 sends a PBU to D-GW1, which replies with a PBA. This PBA MAY include information about the prefix(es) anchored at D-GW1, the parameters needed by D-GW2 to set-up the DLIF mn1dgw1, and the prefixes of local networks reachable via D-GW (if any). Alternatively, this information MAY be obtained using a different approach (such as storing it in the HSS or some other external repository). A bi-directional tunnel between D-GW1 and D-GW2 is set-up, as well as the required routing entries.

8. D-GW2 sets up the DLIF mn1dgw1, aimed at "logically" resembling D-GW1, so MN1 does not detect any change at layer-3. D-GW2 starts sending router advertisements to MN1 through mn1dgw2,

which include the prefix anchored at D-GW1 (prefA::/64) with zero lifetime to deprecate the prefix (or alternatively it MAY include a low Default Router Preference [RFC4191] if communication to this D-GW is still needed in the future). In this way, prefA::MN1 is not preferred for new communications. The RAs MAY also include a Route Information Option (RIO) [RFC4191] with the prefix of localnet, which is the network that is only locally reachable via D-GW1 (e.g., as in the LIPA scenarios considered by the 3GPP), so MN1 picks D-GW1 (the "logical" version of it portrayed by D-GW2) when sending traffic to that network , including the delegated prefix. Traffic using the address prefA::MN1 is received at the interface mn1dgw1 and forwarded via the tunnel with D-GW1, which then forwards it towards its destination. Traffic between MN1 and the network locally reachable via D-GW1 (localnet) is also handled via mn1dgw1 and sent through the tunnel.

```
           +-----+      +-------+      +-------+      +-------------+
           | MN1 |      | D-GW1 |      | D-GW2 |      | CN@Internet |
           +-----+      +-------+      +-------+      +-------------+
              |             |              |                 |
              |  attachment |              |                 |
              |<...........>|set-up of new |                 |
              |  RA@mn1dgw1 |DLIF: mn1dgw1 |                 |
              |<------------|              |                 |
   configures | (prefA::/64)|              |                 |
   prefA::MN1 |             |              |                 |
              |          (traffic using prefA::MN1)          |
              |<------------|------------------------------->|
              | (traffic with local net)   |                 |
              |<----------->|<---->        |                 |
              |             |              |                 |
              |         handover           |                 |
              /............................/set-up of new    |
              |             |   RA@mn1dwg2 |DLIF: mn1dgw2    |
   configures |<----------------------------|                 |
   prefB::MN1 |             |  (prefB::/64)|                 |
   and keeps  |             |         PBU  |                 |
     using    |      tunnel |<-------------|                 |
   prefA::MN1 |      set-up | PBA(mn1dgw1, |                 |
              |             | prefA::/64,  |                 |
              |             | preflocalnet)|                 |
              |             |------------->|set-up of tunnel |
              |             |   RA@mn1dgw1 | & DLIF mn1dgw1  |
              |<----------------------------|                 |
              |             | (prefA::/64, lft=0,            |
              |             |  RIO: preflocalnet)            |
              |             |              |                 |
              |          (traffic using prefB::MN1)          |
              |<--------------------------->|<--------------->|
              |             |              |                 |
              |          (traffic using prefA::MN1)          |
              |<--------------------------->|                 |
              |             |<============>|                 |
              |             |<------------------------------->|
              |             |              |                 |
              | (traffic with local net)   |                 |
              |<--------------------------->|                 |
              |             |<============>|                 |
              |             |<--->         |                 |
              |             |              |                 |
```

Figure 4: D-GW protocol operation

4.3. Message format

This section defines extensions to the Proxy Mobile IPv6 [RFC5213] protocol messages.

4.3.1. Proxy Binding Update

A new flag (D) is included in the Proxy Binding Update to indicate that the Proxy Binding Update is coming from a Distributed Gateway and not from a mobile access gateway. The rest of the Proxy Binding Update format remains the same as defined in [RFC5213].

```
 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
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |            Sequence #         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|A|H|L|K|M|R|P|D| Reserved      |            Lifetime           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                                                               .
.                        Mobility options                       .
.                                                               .

|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Distributed Gateway Flag (D)

The Distributed Gateway Flag is set to indicate to the receiver of the message that the Proxy Binding Update is from a Distributed Gateway.

Mobility Options

Variable-length field of such length that the complete Mobility Header is an integer multiple of 8 octets long. This field contains zero or more TLV-encoded mobility options. The encoding and format of defined options are described in Section 6.2 of [RFC6275]. The distributed gateway MUST ignore and skip any options that it does not understand.

4.3.2. Proxy Binding Acknowledgment

A new flag (D) is included in the Proxy Binding Acknowledgment to indicate that the sender supports operating as a distributed gateway. The rest of the Proxy Binding Acknowledgment format remains the same as defined in [RFC5213].

```
 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
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |   Status      |K|R|P|D| Reser |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Sequence #            |           Lifetime            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                                                               .
.                        Mobility options                       .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Distributed Gateway Flag (D)

The Distributed Gateway Flag is set to indicate that the sender of the message supports operating as a distributed gateway.

Mobility Options

Variable-length field of such length that the complete Mobility Header is an integer multiple of 8 octets long. This field contains zero or more TLV-encoded mobility options. The encoding and format of defined options are described in Section 6.2 of [RFC6275]. The distributed gateway MUST ignore and skip any options that it does not understand.

4.3.3. Anchored Prefix Option

A new Anchored Prefix option is defined for use with the Proxy Binding Update and Proxy Binding Acknowledgment messages exchanged between distributed gateways. This option is used for exchanging the mobile node’s prefix anchored at the anchoring D-GW. There can be multiple Anchored Prefix options present in the message.

The Anchored Prefix Option has an alignment requirement of 8n+4. Its format is as follows:

```
 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      |   Reserved    | Prefix Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                        Anchored Prefix                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

To be assigned by IANA.

Length

8-bit unsigned integer indicating the length of the option in octets, excluding the type and length fields. This field MUST be set to 18.

Reserved

This field is unused for now. The value MUST be initialized to 0 by the sender and MUST be ignored by the receiver.

Prefix Length

8-bit unsigned integer indicating the prefix length of the IPv6 prefix contained in the option.

Anchored Prefix

A sixteen-byte field containing the mobile node’s IPv6 Anchored Prefix.

4.3.4. Local Prefix Option

A new Local Prefix option is defined for use with the Proxy Binding Update and Proxy Binding Acknowledgment messages exchanged between distributed gateways. This option is used for exchanging a prefix of a local network that is only reachable via the anchoring D-GW. There can be multiple Local Prefix options present in the message.

The Local Prefix Option has an alignment requirement of 8n+4. Its format is as follows:

```
 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      |   Reserved    | Prefix Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                          Local Prefix                         +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

To be assigned by IANA.

Length

8-bit unsigned integer indicating the length of the option in octets, excluding the type and length fields. This field MUST be set to 18.

Reserved

This field is unused for now. The value MUST be initialized to 0 by the sender and MUST be ignored by the receiver.

Prefix Length

8-bit unsigned integer indicating the prefix length of the IPv6 prefix contained in the option.

Local Prefix

A sixteen-byte field containing the IPv6 Local Prefix.

4.3.5. DLIF Link-local Address Option

A new DLIF Link-local Address option is defined for use with the Proxy Binding Update and Proxy Binding Acknowledgment messages exchanged between distributed gateways. This option is used for exchanging the link-local address of the DLIF to be configured on the

serving D-GW so it resembles the DLIF configured on the anchoring D-GW.

The DLIF Link-local Address option has an alignment requirement of 8n+6. Its format is as follows:

```
 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     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                  DLIF Link-local Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

To be assigned by IANA.

Length

8-bit unsigned integer indicating the length of the option in octets, excluding the type and length fields. This field MUST be set to 16.

DLIF Link-local Address

A sixteen-byte field containing the link-local address of the logical interface.

4.3.6. DLIF Link-layer Address Option

A new DLIF Link-layer Address option is defined for use with the Proxy Binding Update and Proxy Binding Acknowledgment messages exchanged between distributed gateways. This option is used for exchanging the link-layer address of the DLIF to be configured on the serving D-GW so it resembles the DLIF configured on the anchoring D-GW.

The format of the DLIF Link-layer Address option is shown below. Based on the size of the address, the option MUST be aligned appropriately, as per mobility option alignment requirements specified in [RFC6275].

```
     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     |          Reserved             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                    DLIF Link-layer Address                    +
.                             ...                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

To be assigned by IANA.

Length

8-bit unsigned integer indicating the length of the option in octets, excluding the type and length fields.

Reserved

This field is unused for now. The value MUST be initialized to 0 by the sender and MUST be ignored by the receiver.

DLIF Link-layer Address

A variable length field containing the link-layer address of the logical interface to be configured on the serving distributed gateway.

The content and format of this field (including byte and bit ordering) is as specified in Section 4.6 of [RFC4861] for carrying link-layer addresses. On certain access links, where the link-layer address is not used or cannot be determined, this option cannot be used.

5. Simultaneous anchoring of multiple flows (multiple operators)

An MN may roam between D-GWs that do not belong to the same operator, and therefore might end up having multiple simultaneous flows, anchored at different operators. Since dynamically setting up tunnels between different operators (i.e., between D-GWs belonging to different operators) is usually not supported, a solution should be devised to ensure session continuity in this scenario, even if it is at the cost of a sub-optimal routing.

In this section we describe the required extensions to support inter-domain operation. The basic solution consists in using a centralized LMA (usually located in the home domain) as top-level anchor to guarantee session continuity when crossing operator borders. We assume that the necessary roaming agreements are in place in order to support setting up tunnels between the LMA located at the home domain of the MN and the visited D-GWs.

```
  +-----------------------------------------------------+
 (                  Home Operator's core                 )
 (                                                       )
 (                      +-----+                          )
 (                     /| LMA |\                         )
 (                    //+-----+\\                        )
 (                   //         \\                       )
  +------------------//-----------\\------------------+
         . (tunnel) //             \\ (tunnel) .
         .         //               \\         .
         .        //                 \\        .
         .       //                   \\       .
    +---------------+               +---------------+
    |   IP  stack   |               |   IP  stack   |
    +---------------+               +-------+-------+
    |    mn1dgw1    |--+ (DLIFs) +--|mn1dgw1|mn1dgw2|--+
    +---------------+  |         |  +-------+-------+  |
    | phy interface |  |         |  | phy interface |  |
    +---------------+  |         |  +---------------+  |
     D-GW1@OperatorA  (o)       (o)  D-GW2@OperatorB  (o)
                                   x                 x
                                     x             x
                        prefA::/64     x         x   prefB::/64
                       (AdvPrefLft=0)     x     x
                                            (o)
                                             |
                                          +-----+
                              prefA::MN1  | MN1 |  prefB::MN1
                             (deprecated) +-----+
```

Figure 5: Simultaneous anchoring of multiple flows across multiple operators

Figure 5 shows an example of the inter-domain operation. MN1 initially attaches to D-GW1 (which belongs to OperatorA), and configures prefA::MN1 address out of one prefix anchored at D-GW1 (prefA::/64). If MN1 moves to D-GW2, which is managed by OperatorB, tunnels need to be established via the centralized LMA at the MN1's operators core, since we assume that no direct tunneling is possible between D-GWs belonging to different operators. In this case, D-GW3

establishes one tunnel with the centralized LMA to send/receive traffic using prefA::/64. From the point of view of D-GW2, the operation is just as if the LMA was the D-GW anchoring this prefix. Analogously, the LMA establishes one tunnel with D-GW1 (from the point of view of D-GW1, the LMA is the current serving D-GW of MN1). Regarding the signaling, it is similar to the intra-operator scenario, though in this case the PBU/PBA sequence is performed twice, once between D-GW2 and the LMA, and another one between the LMA and D-GW1 (i.e., because two different tunnels are created).

6. IANA Considerations

This document defines new mobility options that require IANA actions.

7. Security Considerations

The protocol extensions defined in this document share the same security concerns of Proxy Mobile IPv6 [RFC5213]. It is recommended that the signaling messages, Proxy Binding Update and Proxy Binding Acknowledgment, exchanged between the distributed gateways, or between a distributed gateway and a centralized local mobility anchor, are protected using IPsec using the established security association between them. This essentially eliminates the threats related to the impersonation of a distributed gateway or the local mobility anchor.

8. References

8.1. Normative References

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

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <http://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, <http://www.rfc-editor.org/info/rfc4861>.

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

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

8.2. Informative References

[commag.dmm-standards]
Zuniga, JC., Bernardos, CJ., de la Oliva, A., Melia, T., Costa, R., and A. Reznik, "Distributed mobility management: a standards landscape", IEEE Communications Magazine Vol. 51, No. 3, March 2013.

[I-D.bernardos-dmm-pmip]
Bernardos, C., Oliva, A., and F. Giust, "A PMIPv6-based solution for Distributed Mobility Management", draft-bernardos-dmm-pmip-08 (work in progress), March 2017.

[I-D.ietf-dmm-distributed-mobility-anchoring]
Chan, A., Wei, X., Lee, J., Jeon, S., Petrescu, A., and F. Templin, "Distributed Mobility Anchoring", draft-ietf-dmm-distributed-mobility-anchoring-05 (work in progress), May 2017.

[I-D.seite-dmm-dma]
Seite, P., Bertin, P., and J. Lee, "Distributed Mobility Anchoring", draft-seite-dmm-dma-07 (work in progress), February 2014.

[RFC7333] Chan, H., Ed., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", RFC 7333, DOI 10.17487/RFC7333, August 2014, <http://www.rfc-editor.org/info/rfc7333>.

[RFC7429] Liu, D., Ed., Zuniga, JC., Ed., Seite, P., Chan, H., and CJ. Bernardos, "Distributed Mobility Management: Current Practices and Gap Analysis", RFC 7429, DOI 10.17487/RFC7429, January 2015, <http://www.rfc-editor.org/info/rfc7429>.

Appendix A. Comparison with Requirement document

In this section we descrbe how our solution addresses the DMM requirements listed in [RFC7333].

A.1. Distributed mobility management

"IP mobility, network access solutions, and forwarding solutions provided by DMM MUST enable traffic to avoid traversing a single mobility anchor far from the optimal route."

In our solution, the anchoring D-GW is responsible to handle the mobility for those IP flows started when the MN is attached to it. As long as the MN remains connected to the anchoring D-GW's access links, the IP packets of such flows can benefit from the optimal path. When the MN moves to another D-GW, the path becomes non-optimal for ongoing flows, but newly started IP sessions are forwarded by the serving D-GW through the optimal path.

A.2. Bypassable network-layer mobility support for each application session

"DMM solutions MUST enable network-layer mobility, but it MUST be possible for any individual active application session (flow) to not use it. Mobility support is needed, for example, when a mobile host moves and an application cannot cope with a change in the IP address. Mobility support is also needed when a mobile router changes its IP address as it moves together with a host and, in the presence of ingress filtering, an application in the host is interrupted. However, mobility support at the network layer is not always needed; a mobile node can often be stationary, and mobility support can also be provided at other layers. It is then not always necessary to maintain a stable IP address or prefix for an active application session."

The solution operates at the IP layer, hence upper layers are totally transparent to the mobility operations. In particular, ongoing IP sessions are not disrupted after a change of access network. The routability of the old address is ensured by the IP tunnel with the anchoring D-GW. New IP sessions are started with the new address. From the application's perspective, those processes which sockets are bound to a unique IP address do not suffer any impact. For the other applications, the sockets bound to the old address are preserved, whereas next sockets use the new address.

Additionally, the use of the DLIF makes easier to implement more complex policies regarding how traffic is forwarded at the D-GW.

A.3. IPv6 deployment

"DMM solutions SHOULD target IPv6 as the primary deployment environment and SHOULD NOT be tailored specifically to support IPv4,

particularly in situations where private IPv4 addresses and/or NATs are used."

The solution targets IPv6 only.

A.4. Existing mobility protocols

"A DMM solution MUST first consider reusing and extending IETF standard protocols before specifying new protocols."

The is derived from the operations and messages specified in [RFC5213].

A.5. Coexistence with deployed networks/hosts and operability across different networks

"A DMM solution may require loose, tight, or no integration into existing mobility protocols and host IP stacks. Regardless of the integration level, DMM implementations MUST be able to coexist with existing network deployments, end hosts, and routers that may or may not implement existing mobility protocols. Furthermore, a DMM solution SHOULD work across different networks, possibly operated as separate administrative domains, when the needed mobility management signaling, forwarding, and network access are allowed by the trust relationship between them."

The solution can be extended to provide a fallback mechanism to operate as legacy Proxy Mobile IPv6. It is necessary to instruct D-GWs to always establish a tunnel with the same anchoring D-GW, working as LMA.

A.6. Operation and management considerations

"A DMM solution needs to consider configuring a device, monitoring the current operational state of a device, and responding to events that impact the device, possibly by modifying the configuration and storing the data in a format that can be analyzed later.

The proposed solution can re-use existing mechanisms defined for the operation and management of Proxy Mobile IPv6.

A.7. Security considerations

"A DMM solution MUST support any security protocols and mechanisms needed to secure the network and to make continuous security improvements. In addition, with security taken into consideration early in the design, a DMM solution MUST NOT introduce new security

risks or amplify existing security risks that cannot be mitigated by existing security protocols and mechanisms."

The proposed solution does not specify a security mechanism, given that the same mechanism for PMIPv6 can be used.

A.8. Multicast

"DMM SHOULD enable multicast solutions to be developed to avoid network inefficiency in multicast traffic delivery."

This solution in its current version does not specify any support for multicast traffic, which is left for study in future versions.

Appendix B. Implementation experience

The DLIF concept can be easily implemented using features that are usually available on several OSs. Among the possible mechanisms that can be used to do it, the Linux macvlan support allows the creation of different logical interfaces over the same physical one. Each logical interface appears as a regular interface to the Linux OS (which can be configured normally), and it supports configuring the MAC address exposed by the logical interface. The destination MAC address is used by the OS to decide which logical interface (configured on top of a physical interface) is in charge of processing an incoming L2 frame.

The EU FP7 MEDIEVAL project implemented a prototype of the DLIF concept using the Linux macvlan support, the radvd daemon, the Linux Advanced Routing and Traffic Control features and the standard iproute2 collection of utilities:

- The macvlan support enables iproute2 tools to be able to create, destroy and configure DLIFs on demand over a single physical interface. One of the important features that needs to be configured is the logical MAC address exposed by the DLIF, as well as the IPv6 addresses, as they should remain the same regardless of the serving D-GW where the DLIF is configured.

- Since the distributed logical interfaces created using the macvlan support appear as regular network interfaces, they can be used normally in the radvd configuration file. Them, by dynamically modifying the radvd configuration file and reloading it, we can control the router advertisements sent to each MN (e.g., advertizing new IPv6 prefixes, deprecating prefixes anchored at other serving D-GWs, announcing RFC 4191 specific routes or changing router preferences).

- o Each time a DLIF is created, it is also needed to properly configure source-based IPv6 routes, as well as tunnels (in case of handover). This is supported by the Linux Advanced Routing and Traffic Control features.

- o Last, but not least, current Linux kernels support the configuration of RFC 4191 specific routes (by processing Route Information Options contained in RAs). The kernel support can be easily enabled by using the net.conf.ipv6.*.accept_ra_rt_info_max_plen kernel configuration parameter.

The DLIF concept is implemented by the Open Distributed Mobility Management (ODMM) project (http://www.odmm.net/), as part of the Mobility Anchors Distribution for PMIPv6 (MAD-PMIPv6). The ODMM platform is intended to foster DMM development and deployment, by serving as a framework to host open source implementations.

Appendix C. Public demonstrations

The DLIF concept has been demonstrated, together with the network-based DMM solution described in [I-D.bernardos-dmm-pmip], during the 83rd IETF in Paris (March 2012) and the 87th IETF in Berlin (August 2013).

The first demo showcased a scenario composed of three "anchor routers", a "centralized LMA" for control plane, a "mobile node" and two "correspondent nodes" (one of them being a legacy IPv6 camera). The mobile node could move between the different anchor routers, getting a different locally anchor IPv6 address at each location, and being the reachability of each address maintained.

In the second demo, integration with content delivery nodes (CDNs) was also shown, showcasing the advantages that the use of a DMM solution brings to this popular scenario. These concepts were further explored in the EU project MEDIEVAL.

Authors' Addresses

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Juan Carlos Zuniga
SIGFOX
425 rue Jean Rostand
Labege 31670
France

Email: j.c.zuniga@ieee.org
URI: http://www.sigfox.com/

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 12, 2018

CJ. Bernardos
A. de la Oliva
UC3M
F. Giust
NEC

September 8, 2017

A PMIPv6-based solution for Distributed Mobility Management
draft-bernardos-dmm-pmip-09

Abstract

The number of mobile users and their traffic demand is expected to be ever-increasing in future years, and this growth can represent a limitation for deploying current mobility management schemes that are intrinsically centralized, e.g., Mobile IPv6 and Proxy Mobile IPv6. For this reason it has been waved a need for distributed and dynamic mobility management approaches, with the objective of reducing operators' burdens, evolving to a cheaper and more efficient architecture.

This draft describes multiple solutions for network-based distributed mobility management inspired by the well known Proxy Mobile IPv6.

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 RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Current IP mobility solutions, standardized with the names of Mobile IPv6 [RFC6275], or Proxy Mobile IPv6 [RFC5213], just to cite the two most relevant examples, offer mobility support at the cost of handling operations at a cardinal point, the mobility anchor, and burdening it with data forwarding and control mechanisms for a great amount of users. As stated in [RFC7333], centralized mobility solutions are prone to several problems and limitations: longer (sub-optimal) routing paths, scalability problems, signaling overhead (and most likely a longer associated handover latency), more complex network deployment, higher vulnerability due to the existence of a potential single point of failure, and lack of granularity on the mobility management service (i.e., mobility is offered on a per-node basis, not being possible to define finer granularity policies, as for example per-application).

The purpose of Distributed Mobility Management is to overcome the limitations of the traditional centralized mobility management [RFC7333] [RFC7429]; the main concept behind DMM solutions is indeed bringing the mobility anchor closer to the MN. Following this idea, in our proposal, the central anchor is moved to the edge of the network, being deployed in the default gateway of the mobile node. That is, the first elements that provide IP connectivity to a set of MNs are also the mobility managers for those MNs. In the following, we will call these entities Mobility Anchor and Access Routers (MAARs).

This document focuses on network-based DMM, hence the starting point is making PMIPv6 working in a distributed manner [RFC7429]. In our proposal, as in PMIPv6, mobility is handled by the network without the MNs involvement, but, differently from PMIP, when the MN moves from one access network to another, it also changes anchor router, hence requiring signaling between the anchors to retrieve the MN’s previous location(s). Also, a key-aspect of network-based DMM, is that a prefix pool belongs exclusively to each MAAR, in the sense that those prefixes are assigned by the MAAR to the MNs attached to it, and they are routable at that MAAR.

In the following, we consider two main approaches to design our DMM solutions:

- Partially distributed schemes, where the data plane only is distributed among access routers similar to MAGs, whereas the control plane is kept centralized towards a cardinal node used as information store, but relieved from any route management and MN’s data forwarding task.

- o Fully distributed schemes, where both data and control planes are distributed among the access routers.

2. Terminology

The following terms used in this document are defined in the Proxy Mobile IPv6 specification [RFC5213]:

Local Mobility Anchor (LMA)

Mobile Access Gateway (MAG)

Mobile Node (MN)

Binding Cache Entry (BCE)

Proxy Care-of Address (P-CoA)

Proxy Binding Update (PBU)

Proxy Binding Acknowledgement (PBA)

The following terms are defined and used in this document:

MAAR (Mobility Anchor and Access Router). First hop router where the mobile nodes attach to. It also plays the role of mobility manager for the IPv6 prefixes it anchors, running the functionalities of PMIP’s MAG and LMA.

CMD (Central Mobility Database). Node that stores the BCEs allocated for the MNs in the mobility domain.

P-MAAR (Previous MAAR). MAAR which was previously visited by the MN and is still involved in an active flow using an IPv6 prefix it has advertised to the MN (i.e., MAAR where that IPv6 prefix is anchored). There might be multiple P-MAARs for an MN’s mobility session.

S-MAAR (Serving MAAR). MAAR which the MN is currently attached to.

3. Partially distributed solution

The following solution consists in de-coupling the entities that participates in the data and the control planes: the data plane becomes distributed and managed by the MAARs near the edge of the network, while the control plane, besides on the MAARs, relies on a central entity called Central Mobility Database (CMD). In the

proposed architecture, the hierarchy present in PMIP between LMA and MAG is preserved, but with the following substantial variations:

- o The LMA is relieved from the data forwarding role, only the Binding Cache and its management operations are maintained. Hence the LMA is renamed into Central Mobility Database (CMD). Also, the CMD is able to send and parse both PBU and PBA messages.

- o The MAG is enriched with the LMA functionalities, hence the name Mobility Anchor and Access Router (MAAR). It maintains a local Binding Cache for the MNs that are attached to it and it is able to send and parse PBU and PBA messages.

- o The binding cache will have to be extended to include information regarding previous MAARs where the mobile node was anchored and still retains active data sessions, see Appendix B for further details.

- o Each MAAR has a unique set of global prefixes (which are configurable), that can be allocated by the MAAR to the MNs, but must be exclusive to that MAAR, i.e. no other MAAR can allocate the same prefixes.

The MAARs leverage on the Central Mobility Database (CMD) to access and update information related to the MNs, stored as mobility sessions; hence, a centralized node maintains a global view on the status of the network. The CMD is queried whenever a MN is detected to join/leave the mobility domain. It might be a fresh attachment, a detachment or a handover, but as MAARs are not aware of past information related to a mobility session, they contact the CMD to retrieve the data of interest and eventually take the appropriate action. The procedure adopted for the query and the messages exchange sequence might vary to optimize the update latency and/or the signaling overhead. Here is presented one method for the initial registration, and three different approaches to update the mobility sessions using PBUs and PBAs. Each approach assigns a different role to the CMD:

- o The CMD is a PBU/PBA relay;

- o The CMD is only a MAAR locator;

- o The CMD is a PBU/PBA proxy.

3.1. Initial registration

Upon the MN's attachment to a MAAR, say MAAR1, if the MN is authorized for the service, an IPv6 global prefix belonging to the MAAR's prefix pool is reserved for it (Pref1) into a temporal Binding Cache Entry (BCE) allocated locally. The prefix is sent in a [RFC5213] PBU with the MN's Identifier (MN-ID) to the CMD, which, since the session is new, stores a permanent BCE containing as main fields the MN-ID, the MN's prefix and MAAR1's address as Proxy-CoA. The CMD replies to MAAR1 with a PBA including the usual options defined in PMIP/RFC5213, meaning that the MN's registration is fresh and no past status is available. MAAR1 definitely stores the temporal BCE previously allocated and unicasts a Router Advertisement (RA) to the MN including the prefix reserved before, that can be used by the MN to configure an IPv6 address (e.g., with stateless auto-configuration). The address is routable at the MAAR, in the sense that it is on the path of packets addressed to the MN. Moreover, the MAAR acts as plain router for those packets, as no encapsulation nor special handling takes place. Figure 1 illustrates this scenario.

```
  +-----+      +---+                +--+
  |MAAR1|      |CMD|                |CN|
  +-----+      +---+                +*-+
     |           |                   *
    MN           |                   *     +---+
  attach.        |               *****    _|CMD|_
detection        |         flow1 *       / +-+-+ \
     |           |               *      /    |    \
 local BCE       |               *     /     |     \
 allocation      |               *    /      |      \
     |--- PBU -->|           +---*-+-'    +--+--+    '+-----+
     |          BCE          |   * |      |     |     |     |
     |        creation       |MAAR1+------+MAAR2+-----+MAAR3|
     |<-- PBA ---|           |   * |      |     |     |     |
 local BCE       |           +---*-+      +-----+     +-----+
 finalized       |               *
     |           |         Pref1 *
     |           |              +*-+
     |           |              |MN|
     |           |              +--+

  Operations sequence                  Packets flow
```

Figure 1: First attachment to the network

3.2. The CMD as PBU/PBA relay

When the MN moves from its current access and associates to MAAR2 (now the S-MAAR), MAAR2 reserves another IPv6 prefix (Pref2), it stores a temporal BCE, and it sends a plain PBU to the CMD for registration. Upon PBU reception and BC lookup, the CMD retrieves an already existing entry for the MN, binding the MN-ID to its former location; thus, the CMD forwards the PBU to the MAAR indicated as Proxy CoA (MAAR1), including a new mobility option to communicate the S-MAAR's global address to MAAR1, defined as Serving MAAR Option in Section 3.6.2. The CMD updates the P-CoA field in the BCE related to the MN with the S-MAAR's address.

Upon PBU reception, MAAR1 can install a tunnel on its side towards MAAR2 and the related routes for Pref1. Then MAAR1 replies to the CMD with a PBA (including the option mentioned before) to ensure that the new location has successfully changed, containing the prefix anchored at MAAR1 in the Home Network Prefix option. The CMD, after receiving the PBA, updates the BCE populating an instance of the P-MAAR list. The P-MAAR list is an additional field on the BCE that contains an element for each P-MAAR involved in the MN's mobility session. The list element contains the P-MAAR's global address and the prefix it has delegated (see Appendix B for further details). Also, the CMD send a PBA to the new S-MAAR, containing the previous Proxy-CoA and the prefix anchored to it embedded into a new mobility option called Previous MAAR Option (defined in Section 3.6.1), so that, upon PBA arrival, a bi-directional tunnel can be established between the two MAARs and new routes are set appropriately to recover the IP flow(s) carrying Pref1.

Now packets destined to Pref1 are first received by MAAR1, encapsulated into the tunnel and forwarded to MAAR2, which finally delivers them to their destination. In uplink, when the MN transmits packets using Pref1 as source address, they are sent to MAAR2, as it is MN's new default gateway, then tunneled to MAAR1 which routes them towards the next hop to destination. Conversely, packets carrying Pref2 are routed by MAAR2 without any special packet handling both for uplink and downlink. The procedure is depicted in Figure 2.

```
   +-----+      +---+      +-----+           +--+            +--+
   |MAAR1|      |CMD|      |MAAR2|           |CN|            |CN|
   +-----+      +---+      +-----+           +*-+            +*-+
      |           |           |               *               *
      |           |          MN               *     +---+     *
      |           |        attach.        *****    _|CMD|_    *
      |           |          det.   flow1 *       / +-+-+ \   *flow2
      |           |<-- PBU ---|           *      /    |    \  *
      |          BCE          |           *     /     | *******
      |        check+         |           *    /      | *    \
      |        update         |       +---*-+-?    +--+-*+    '+-----+
      |<-- PBU*---|           |       |   * |      |    *|     |     |
   route          |           |       |MAAR1|______|MAAR2+-----+MAAR3|
   update         |           |       |   **(______)**  *|     |     |
      |--- PBA*-->|           |       +-----+      +-*--*+     +-----+
      |         BCE           |                      *  *
      |        update         |                Pref1 *  *Pref2
      |           |--- PBA*-->|                     +*--*+
      |           |         route         ---move-->|*MN*|
      |           |         update                  +----+

         Operations sequence                  Data Packets flow
   PBU/PBA Messages with * contain
           a new mobility option
```

Figure 2: Scenario after a handover, CMD as relay

For next MN's movements the process is repeated except for the number of P-MAARs involved, that rises accordingly to the number of prefixes that the MN wishes to maintain. Indeed, once the CMD receives the first PBU from the new S-MAAR, it forwards copies of the PBU to all the P-MAARs indicated in the BCE as current P-CoA (i.e., the MAAR prior to handover) and in the P-MAARs list. They reply with a PBA to the CMD, which aggregates them into a single one to notify the S-MAAR, that finally can establish the tunnels with the P-MAARs.

It should be noted that this design separates the mobility management at the prefix granularity, and it can be tuned in order to erase old mobility sessions when not required, while the MN is reachable through the latest prefix acquired. Moreover, the latency associated to the mobility update is bound to the PBA sent by the furthest P-MAAR, in terms of RTT, that takes the longest time to reach the CMD. The drawback can be mitigated introducing a timeout at the CMD, by which, after its expiration, all the PBAs so far collected are transmitted, and the remaining are sent later upon their arrival.

3.3. The CMD as MAAR locator

The handover latency experienced in the approach shown before can be reduced if the P-MAARs are allowed to signal directly their information to the new S-MAAR. This procedure reflect what was described in Section 3.2 up to the moment the P-MAAR receives the PBU with the P-MAAR option. At that point a P-MAAR is aware of the new MN's location (because of the S-MAAR's address in the S-MAAR option), and, besides sending a PBA to the CMD, it also sends a PBA to the S-MAAR including the prefix it is anchoring. This latter PBA does not need to include new options, as the prefix is embedded in the HNP option and the P-MAAR's address OS taken from the message's source address. The CMD is relieved from forwarding the PBA to the S-MAAR, as the latter receives a copy directly from the P-MAAR with the necessary information to build the tunnels and set the appropriate routes. In Figure 3 is illustrated the new messages sequence, while the data forwarding is unaltered.

```
+-----+      +---+      +-----+           +--+            +--+
|MAAR1|      |CMD|      |MAAR2|           |CN|            |CN|
+-----+      +---+      +-----+           +*-+            +*-+
   |           |           |                *               *
   |           |          MN                *     +---+     *
   |           |        attach.        *****    _|CMD|_    *
   |           |          det.   flow1 *       / +-+-+ \   *flow2
   |           |<-- PBU ---|           *      /    |    \  *
   |          BCE          |           *     /     | *******
   |        check+         |           *    /      | *    \
   |        update         |       +---*-+-?    +--+-*+    '+-----+
   |<-- PBU*---|           |       |   * |      |    *|     |     |
route          |           |       |MAAR1|______|MAAR2+-----+MAAR3|
update         |           |       |   **(______)**  *|     |     |
   |--------- PBA -------->|       +-----+      +-*--*+     +-----+
   |--- PBA*-->|         route                     *  *
   |          BCE        update             Pref1 *  *Pref2
   |         update        |                    +*--*+
   |           |           |           ---move-->|*MN*|
   |           |           |                     +----+

        Operations sequence                  Data Packets flow
PBU/PBA Messages with * contain
        a new mobility option
```

Figure 3: Scenario after a handover, CMD as locator

3.4. The CMD as MAAR proxy

A further enhancement of previous solutions can be achieved when the CMD sends the PBA to the new S-MAAR before notifying the P-MAARs of the location change. Indeed, when the CMD receives the PBU for the new registration, it is already in possess of all the information that the new S-MAAR requires to set up the tunnels and the routes. Thus the PBA is sent to the S-MAAR immediately after a PBU is received, including also in this case the P-MAAR option. In parallel, a PBU is sent by the CMD to the P-MAARs containing the S-MAAR option, to notify them about the new MN's location, so they receive the information to establish the tunnels and routes on their side. When P-MAARs complete the update, they send a PBA to the CMD to indicate that the operation is concluded and the information are updated in all network nodes. This procedure is obtained from the first one re-arranging the order of the messages, but the parameters communicated are the same. This scheme is depicted in Figure 4, where, again, the data forwarding is kept untouched.

```
+-----+      +---+      +-----+           +--+            +--+
|MAAR1|      |CMD|      |MAAR2|           |CN|            |CN|
+-----+      +---+      +-----+           +*-+            +*-+
   |           |           |               *               *
   |           |          MN               *     +---+     *
   |           |        attach.        *****    _|CMD|_    *
   |           |          det.   flow1 *       / +-+-+ \   *flow2
   |           |<-- PBU ---|           *      /    |    \  *
   |          BCE          |           *     /     | *******
   |        check+         |           *    /      | *    \
   |        update         |       +---*-+-?    +--+-*+    '+-----+
   |<-- PBU*---x--- PBA*-->|       |   * |      |    *|     |     |
route          |         route     |MAAR1|______|MAAR2+-----+MAAR3|
update         |         update    |   **(______)**  *|     |     |
   |--- PBA*-->|           |       +-----+      +-*--*+     +-----+
   |          BCE          |                      *  *
   |         update        |                Pref1 *  *Pref2
   |           |           |                     +*--*+
   |           |           |           ---move-->|*MN*|
   |           |           |                     +----+

        Operations sequence                 Data Packets flow
PBU/PBA Messages with * contain
        a new mobility option
```

Figure 4: Scenario after a handover, CMD as proxy

3.5. De-registration

The de-registration mechanism devised for PMIPv6 is no longer valid in the Partial DMM architecture. This is motivated by the fact that each MAAR handles an independent mobility session (i.e., a single or a set of prefixes) for a given MN, whereas the aggregated session is stored at the CMD. Indeed, when a previous MAAR initiates a de-registration procedure, because the MN is no longer present on the MAAR's access link, it removes the routing state for that (those) prefix(es), that would be deleted by the CMD as well, hence defeating any prefix continuity attempt. The simplest approach to overcome this limitation is to deny an old MAAR to de-register a prefix, that is, allowing only a serving MAAR to de-register the whole MN session. This can be achieved by first removing any layer-2 detachment event, so that de-registration is triggered only when the session lifetime expires, hence providing a guard interval for the MN to connect to a new MAAR. Then, a change in the MAAR operations is required, and at this stage two possible solutions can be deployed:

- o A previous MAAR stops the BCE timer upon receiving a PBU from the CMD containing a "Serving MAAR" option. In this way only the Serving MAAR is allowed to de-register the mobility session, arguing that the MN left definitely the domain.

- o Previous MAARs can, upon BCE expiry, send de-registration messages to the CMD, which, instead of acknowledging the message with a 0 lifetime, send back a PBA with a non-zero lifetime, hence renewing the session, if the MN is still connected to the domain.

The evaluation of these methods is left for future work.

3.6. Message Format

This section defines two Mobility Options to be used in the PBU and PBA messages:

Previous MAAR Option;

Serving MAAR Option.

In the current draft the messages reflect IPv6 format only. IPv4 compatibility will be added in next release.

3.6.1. Previous MAAR Option

This new option is defined for use with the Proxy Binding Acknowledgement messages exchanged by the CMD to a MAAR. This option is used to notify the S-MAAR about the previous MAAR's global address

and the prefix anchored to it. There can be multiple Previous MAAR options present in the message. Its format is as follows:

```
 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    | Prefix Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                     P-MAAR's address                          +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                    Home Network Prefix                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

To be assigned by IANA.

Length

8-bit unsigned integer indicating the length of the option in octets, excluding the type and length fields. This field MUST be set to 34.

Prefix Length

8-bit unsigned integer indicating the prefix length of the IPv6 prefix contained in the option.

Previous MAAR's address

A sixteen-byte field containing the P-MAAR's IPv6 global address.

Home Network Prefix

A sixteen-byte field containing the mobile node's IPv6 Home Network Prefix.

3.6.2. Serving MAAR Option

This new option is defined for use with the Proxy Binding Update and Proxy Binding Acknowledgement messages exchanged between the CMD and a Previous MAAR. This option is used to notify the P-MAAR about the current Serving MAAR's global address. Its format is as follows:

```
 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    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                     S-MAAR's address                          +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

To be assigned by IANA.

Length

8-bit unsigned integer indicating the length of the option in octets, excluding the type and length fields. This field MUST be set to 16.

Serving MAAR's address

A sixteen-byte field containing the S-MAAR's IPv6 global address.

4. Fully distributed solution

In this section we introduce the guidelines to evolve our partially DMM solution into a fully distributed one. We list the key concepts in the following (some of the points are already enforced in previous sections of this document):

- o All MAARs have a pool of global routable IPv6 prefixes to be assigned to MNs on the access link.

- o Any central control entity is removed from the architecture and each MAAR will retain it’s own cache for the mobile nodes directly anchored to it.

- o Both control and data planes are now entirely handled by the MAARs.

Because we aim for a fully distributed approach, the lack of knowledge of other MAARs and their advertised prefixes becomes a serious obstacle. In this particular case, when a MN attaches to a MAAR, there are two main pieces ofi nformation that this MAAR requires to know, to properly assure a mobile node’s mobility and continuity of its data flows: i) if the node has any P-MAARs and their addresses; ii) if it has P-MAARs, which prefixes were advertised by which MAAR.

There are several methods to achieve this:

- o Make before approaches, employing Layer 2 or Layer 3 mechanisms. The target MAAR is known in advance by the current MAAR before handover, hence the mobility context can be transferred.

- o Distributed schemes for MAAR discovery: it can based on a peer-to-peer approach; or it can employ a unicast, multicast or broadcast query system.

- o Explicit notification by the MN. For example, extending the layer three IP address configuration mechanisms (e.g., ND).

- o Other MN to MAAR communication protocol (e.g., IEEE 802.21).

5. IANA Considerations

TBD.

6. Security Considerations

The solution assumes that the nodes are trusted and secure MAAR-to-MAAR communications are in place, for instance re-using the security mechanisms defined for PMIPv6. Thus, the solution does not introduce any new security vulnerability.

7. Acknowledgments

The authors would like to thank Marco Liebsch for his comments and discussion on this document.

8. References

8.1. Normative References

[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>.

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

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

8.2. Informative References

[I-D.bernardos-dmm-distributed-anchoring] Bernardos, C. and J. Zuniga, "PMIPv6-based distributed anchoring", draft-bernardos-dmm-distributed-anchoring-09 (work in progress), May 2017.

[RFC7333] Chan, H., Ed., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", RFC 7333, DOI 10.17487/RFC7333, August 2014, <https://www.rfc-editor.org/info/rfc7333>.

[RFC7429] Liu, D., Ed., Zuniga, JC., Ed., Seite, P., Chan, H., and CJ. Bernardos, "Distributed Mobility Management: Current Practices and Gap Analysis", RFC 7429, DOI 10.17487/RFC7429, January 2015, <https://www.rfc-editor.org/info/rfc7429>.

Appendix A. Comparison with Requirement document

In this section we descrbe how our solution addresses the DMM requirements listed in [RFC7333].

A.1. Distributed mobility management

"IP mobility, network access solutions, and forwarding solutions provided by DMM MUST enable traffic to avoid traversing a single mobility anchor far from the optimal route."

In our solution, a MAAR is responsible to handle the mobility for those IP flows started when the MN is attached to it. As long as the MN remains connected to the MAAR's access links, the IP packets of such flows can benefit from the optimal path. When the MN moves to another MAAR, the path becomes non-optimal for ongoing flows, as they are anchored to the previous MAAR, but newly started IP sessions are forwarded by the new MAAR through the optimal path.

A.2. Bypassable network-layer mobility support for each application session

"DMM solutions MUST enable network-layer mobility, but it MUST be possible for any individual active application session (flow) to not use it. Mobility support is needed, for example, when a mobile host moves and an application cannot cope with a change in the IP address. Mobility support is also needed when a mobile router changes its IP address as it moves together with a host and, in the presence of ingress filtering, an application in the host is interrupted. However, mobility support at the network layer is not always needed; a mobile node can often be stationary, and mobility support can also be provided at other layers. It is then not always necessary to maintain a stable IP address or prefix for an active application session."

Our DMM solution operates at the IP layer, hence upper layers are totally transparent to the mobility operations. In particular, ongoing IP sessions are not disrupted after a change of access network. The routability of the old address is ensured by the IP tunnel with the old MAAR. New IP sessions are started with the new address. From the application's perspective, those processes which sockets are bound to a unique IP address do not suffer any impact. For the other applications, the sockets bound to the old address are preserved, whereas next sockets use the new address.

A.3. IPv6 deployment

"DMM solutions SHOULD target IPv6 as the primary deployment environment and SHOULD NOT be tailored specifically to support IPv4, particularly in situations where private IPv4 addresses and/or NATs are used."

The DMM solution we propose targets IPv6 only.

A.4. Existing mobility protocols

"A DMM solution MUST first consider reusing and extending IETF standard protocols before specifying new protocols."

This DMM solution is derived from the operations and messages specified in [RFC5213].

A.5. Coexistence with deployed networks/hosts and operability across different networks

"A DMM solution may require loose, tight, or no integration into existing mobility protocols and host IP stacks. Regardless of the integration level, DMM implementations MUST be able to coexist with existing network deployments, end hosts, and routers that may or may not implement existing mobility protocols. Furthermore, a DMM solution SHOULD work across different networks, possibly operated as separate administrative domains, when the needed mobility management signaling, forwarding, and network access are allowed by the trust relationship between them"

The partially DMM solution can be extended to provide a fallback mechanism to operate as legacy Proxy Mobile IPv6. It is necessary to instruct MAARs to always establish a tunnel with the same MAAR, working as LMA. The fully DMM solution can be extended as well, but it requires more intervention. The partially DMM solution can be deployed across different domains with trust agreements if the CMDs ot the operators are enabled to transfer context from one node to another. The fully DMM solution works across multiple domains if both solution apply the same signalling scheme.

A.6. Operation and management considerations

"A DMM solution needs to consider configuring a device, monitoring the current operational state of a device, and responding to events that impact the device, possibly by modifying the configuration and storing the data in a format that can be analyzed later.

The proposed solution can re-use existing mechanisms defined for the operation and management of Proxy Mobile IPv6.

A.7. Security considerations

"A DMM solution MUST support any security protocols and mechanisms needed to secure the network and to make continuous security improvements. In addition, with security taken into consideration early in the design, a DMM solution MUST NOT introduce new security risks or amplify existing security risks that cannot be mitigated by existing security protocols and mechanisms."

The proposed solution does not specify a security mechanism, given that the same mechanism for PMIPv6 can be used.

A.8. Multicast

"DMM SHOULD enable multicast solutions to be developed to avoid network inefficiency in multicast traffic delivery."

This solution in its current version does not specify any support for multicast traffic, which is left for study in future versions.

Appendix B. Implementation experience

The network-based DMM solution described in section Section 3.4 is now available at the Open Distributed Mobility Management (ODMM) project (http://www.odmm.net/), under the name of Mobility Anchors Distribution for PMIPv6 (MAD-PMIPv6). The ODMM platform is intended to foster DMM development and deployment, by serving as a framework to host open source implementations.

The MAD-PMIPv6 code is developed in ANSI C from the existing UMIP implementation for PMIP. The most relevant changes with respect to the UMIP original version are related to how to create the CMD and MAAR’s state machines from those of an LMA and a MAG; for this purpose, part of the LMA code was copied to the MAG, in order to send PBA messages and parse PBU. Also, the LMA routing functions were removed completely, and moved to the MAG, because MAARs need to route through the tunnels in downlink (as an LMA) and in uplink (as a MAG).

Tunnel management is hence a relevant technical aspect, as multiple tunnels are established by a single MAAR, which keeps their status directly into the MN’s BCE. Indeed, from the implementation experience it was chosen to create an ancillary data structure as field within a BCE: the data structure is called "MAAR list" and stores the previous MAARs’ address and the corresponding prefix(es) assigned for the MN. Only the CMD and the serving MAAR store this data structure, because the CMD maintains the global MN’s mobility session formed during the MN’s roaming within the domain, and the serving MAAR needs to know which previous MAARs were visited, the prefix(es) they assigned and the tunnels established with them. Conversely, a previous MAAR only needs to know which is the current Serving MAAR and establish a single tunnel with it. For this reason, a MAAR that receives a PBU from the CMD (meaning that the MN attached to another MAAR), first sets up the routing state for the MN’s prefix(es) it is anchoring, then stop the BCE expiry timer and deletes the MAAR list (if present) since it is no longer useful.

In order to have the MN totally unaware of the changes in the access link, all MAARs implement the Distributed Logical Interface (DLIF) concept devised in [I-D.bernardos-dmm-distributed-anchoring]. Moreover, it should be noted that the protocols designed in the

document work only at the network layer to handle the MNs joining or leaving the domain. This should guarantee a certain independency to a particular access technology. The implementation reflects this reasoning, but we argue that an interaction with lower layers produces a more effective attachment and detachment detection, therefore improving the performance, also regarding de-registration mechanisms.

It was chosen to implement the "proxy" solution because it produces the shortest handover latency, but a slight modification on the CMD state machine can produce the first scenario described ("relay") which guarantees a more consistent request/ack scheme between the MAARS. By modifying also the MAAR’s state machine it can be implemented the second solution ("locator").

An early MAD-PMIPv6 implementation was shown during a demo session at the IETF 83rd, in Paris in March 2012. An enhancement version of the prototype has been presented at the 87th IETF meeting in Berlin, July 2013. The updated demo included a use case scenario employing a CDN system for video delivery. More, MAD-PMIPv6 has been extensively used and evaluated within a testbed employing heterogeneous radio accesses within the framework of the MEDIEVAL EU project. MAD-PMIPv6 software is currently part of a DMM test-bed comprising 3 MAARs, one CMD, one MN and a CN. All the machines used in the demos were Linux UBUNTU 10.04 systems with kernel 2.6.32, but the prototype has been tested also under newer systems. This testbed was also used by the iJOIN EU project.

Authors’ Addresses

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Antonio de la Oliva
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 8803
Email: aoliva@it.uc3m.es
URI: http://www.it.uc3m.es/aoliva/

Fabio Giust
NEC Laboratories Europe
NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg D-69115
Germany

Phone: +49 6221 4342216
Email: fabio.giust@neclab.eu

DMM
Internet-Draft
Intended status: Informational
Expires: January 4, 2015

H. Chan, Ed.
Huawei Technologies
K. Pentikousis, Ed.
EICT

July 3, 2014

Enhanced mobility anchoring
draft-chan-dmm-enhanced-mobility-anchoring-00

Abstract

This document initiates the discussion on enhanced mobility anchoring solutions in the context of a distributed mobility management deployment. Such solutions consider the problem of assigning a mobility anchor and a gateway at the initiation of a session. In addition, the mid-session switching of the mobility anchor in a distributed mobility management environment is considered.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 4, 2015.

Table of Contents

1. Introduction

A key requirement in distributed mobility management [I-D.ietf-dmm-requirements] is to enable traffic to avoid traversing single mobility anchor far from the optimal route. Recent developments in research and standardization with respect to future deployment models call for far more flexibility in network function operation and management. For example, the work on service function chaining at the IETF (SFC WG) has already identified a number of use cases for data centers. Although the work in SFC is not primarily concerned with mobile networks, the impact on IP-based mobile networks is not hard to see as by now most hosts connected to the Internet do so over a wireless medium. For instance, as a result of a dynamic re-organization of service chain a non-optimal route between mobile nodes may arise if pme relies solely on centralized mobility management. As discussed earlier in the distributed mobility management working group (DMM WG) this may also occur when the mobile node has moved such that both the mobile node and the correspondent node are far from the mobility anchor via which the traffic is routed.

Motivated by the above-mentioned developments as well as [I-D.ietf-dmm-requirements] we aim with this draft to initiate the discussion on enhanced mobility anchoring. Recall that distributed mobility management solutions do not make use of centrally deployed mobility anchor. As such, an application session SHOULD be able to have its traffic passing from one mobility anchor to another as the mobile node moves, or when changing operation and management (OAM) requirements call for mobility anchor switching, thus avoiding non-optimal routes.

2. Conventions and Terminology

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].

All general mobility-related terms and their acronyms used in this document are to be interpreted as defined in the Mobile IPv6 base specification [RFC6275] and in the Proxy Mobile IPv6 specification [RFC5213]. This includes terms such as mobile node (MN), correspondent node (CN), home agent (HA), home address (HoA), care-of-address (CoA), local mobility anchor (LMA), and mobile access gateway (MAG).

In addition, this document uses the following term:

Home network of an application session (or of an HoA) is the network that has allocated the IP address (HoA) used for the session identifier by the application running in an MN. An MN may be running multiple application sessions, and each of these sessions can have a different home network.

3. Enhanced anchor switching

In this section we consider mid-session mobility anchor switching for two cases. First we discuss the case where the mobile node moves from one subnet to another, and then we discuss the case where the node moves to a different network. Note that although the cases are described with traditional (read: physical) node mobility in mind, the proposed mechanism can be triggered for other operational reasons, such as the redefinition of a service chain graph, due to mechanisms which indicate that by relocating the mobility anchor for certain sessions energy and other operation expenditure can be reduced, or due to emergency situations, such as physical catastrophes.

3.1. Anchor switching between subnets

First we consider the situation illustrated in Fig. 1: The mobile node (M) moves from Subnet 1 to Subnet 2. Each of the Network A Subnets (1, 2, and so on) owns a block of IP addresses. In each subnet, the corresponding access router (AR1, AR2, ...) advertises the routes for the block of addresses of that subnet.

```
                +----------+
                | Network A|
                |Controller|
                +----------+
     +---+         +---+         +---+
     |AR1|         |AR2|         |AR3|
   +--------+    +--------+    +--------+
   |Subnet 1|    |Subnet 2|    |Subnet 3|  ....
   +--------+    +--------+    +--------+

     +----+        +----+
     | M  | =====> | M  |
     |with|        |with|
     |IP1 |        |IP2 |
     |    |        |IP1 |
     +----+        +----+
```

Figure 1. Movement of M from Subnet 1 to Subnet 2.

Before moving, M is allocated an IP address IP1 from Subnet 1, and it may run network applications using this IP address.

As M moves to Subnet 2, it obtains a new IP address IP2 from Subnet 2. The applications that can handle a change of IP address will use the address IP2 [I-D.seite-dmm-dma]. Other ongoing applications that cannot survive an IP address change will need to continue using IP11 to maintain session continuity. A mobility management protocol may be used to enable M to use the address IP1 belonging to Subnet 1.

The AR1 access router in Subnet 1 may delegate the IP address (IP1) to the access router AR2 in Subnet 2. AR2 will then advertise IP1 so that the routing tables in Network A will be updated and packets destined to IP1 will be routed to Subnet 2.

Relying on earlier routing table update mechanisms with a distributed routing protocol may not be fast enough to meet the requirement for a short handover delay. In the case where a control and data plane separation model is followed, a logically centralized mechanism can perform the forwarding table update faster. For example, we can consider the use of I2RS mechanisms or the possibility to employ NETCONF [RFC6241] for reconfiguring AR2.

Alternatively, a tunneling mobility management protocol such as MIPv6 [RFC6275] or PMIPv6 [RFC5213] may be used initially [Paper-Distributed.Mobility.PMIP] to enable M to use the IP address IP1 while IP1 still belongs to Subnet 1. The route may not be optimized initially, but this is a good tradeoff so that anchor switching can

take place. After anchor switching and its subsequent forwarding table update have been completed, packets destined to IP1 will be routed directly towards M.

The address delegation of IP1 from Subnet 1 to Subnet 2 may timeout unless there is request to renew it before it expires. When all applications using IP1 in M have been terminated, there will be no longer need for using IP1 in Subnet 2. If there are still such applications running when the address delegation is about to timeout, the mobile node may signal with AR1 to request renewal of address delegation.

3.2. Anchor switching between networks

Fig. 2 illustrates the movement of a mobile node (M) from Subnet a1 of Network A to Subnet b2 of Network B. In this case, each Network (A, B, and so on) owns the aggregate of IP addresses blocks for its subnets. The corresponding gateway routers (GWa, GWb, ...) may run an IBGP among them, and each advertises the aggregate of IP addresses for its subnets.

```
   +---+           +---+           +---+
   |GWa|           |GWb|           |GWc|
+----------+    +----------+    +----------+
|Network A |    |Network B |    |Network C |   ....
|Controller|    |Controller|    |Controller|
+----------+    +----------+    +----------+

  +----+          +----+
  |ARa1|          |ARb2|
+---------+     +---------+
|Subnet a1|     |Subnet b2|     ....
+---------+     +---------+

  +-----+         +-----+
  |  M  |  =====> |  M  |
  | with|         | with|
  |IPa11|         |IPb21|
  |     |         |IPa11|
  +-----+         +-----+
```

Figure 1. Movement of M from Subnet a1 of Network A to Subnet b2 of Network B.

Before moving, M is allocated the IP address IPa11 from Subnet a1 of Network A, and it may run network applications using this IP address.

As M moves to Subnet b2, it obtains a new IP address IPb21 from Subnet b2 of Network B. The applications that can handle a change of IP address will use this new address IPb21. Other applications with ongoing sessions that cannot survive an IP address change will need to continue using IPa11 to maintain session continuity. A mobility management protocol may be used to enable M to use the address IPa11 belonging to Subnet a1 in Network A.

As the access router ARa1 in Subnet a1 may delegate the address IPa11 to the access router ARb2 in Subnet b2, the gateways GWa, GWb, ... also need to update the routing information so that GWb will then advertise IPa11 so that the routing tables in GWa, GWb, ... will update and packets destined to IPa11 will be routed to Network B.

The routing table update between the gateways MAY be accomplished using IS-IS. In scenarios where the control plane and the data plane for these gateways are separate, and there is a controller for these gateways, a centralized routing protocol can also perform the forwarding table update for these gateways.

Optionally, a tunneling mobility management protocol such as MIPv6 [RFC6275] or PMIPv6 [RFC5213] may be used to initially enable M to use the address IPa11 while IPa11 still belongs to Subnet a1 of Network A. Although such a route may not be optimized initially, it enables anchor switching to take place. After anchor switching and its subsequent forwarding table update have been completed, the packets destined to IPa11 will be routed directly towards M.

The address delegation of IPa11 from Subnet a1 to Subnet b2 may timeout unless there is request to renew it before it expires. When the applications uisng IPa11 in M have all been terminated, there will be no longer need for using IPa11 in Subnet b2. If there are still such applications running when the address delegation is about to timeout, the mobile node may signal with ARa1 to request renewal of address delegation.

4. Security Considerations

TBD

5. IANA Considerations

This document presents no IANA considerations.

6. References

6.1. Normative References

[I-D.ietf-dmm-requirements]
Chan, A., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", draft-ietf-dmm-requirements-17 (work in progress), June 2014.

[I-D.seite-dmm-dma]
Seite, P., Bertin, P., and J. Lee, "Distributed Mobility Anchoring", draft-seite-dmm-dma-07 (work in progress), February 2014.

[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.

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

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

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

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

6.2. Informative References

[Paper-Distributed.Mobility.PMIP]
Chan, H., "Proxy Mobile IP with Distributed Mobility Anchors", Proceedings of GlobeCom Workshop on Seamless Wireless Mobility, December 2010.

[Paper-Distributed.Mobility.Review]
Chan, H., Yokota, H., Xie, J., Seite, P., and D. Liu, "Distributed and Dynamic Mobility Management in Mobile Internet: Current Approaches and Issues", February 2011.

Authors' Addresses

H Anthony Chan
Huawei Technologies
5340 Legacy Dr. Building 3
Plano, TX 75024
USA

Email: h.a.chan@ieee.org

Kostas Pentikousis
EICT
EUREF-Campus Haus 13
Torgauer Strasse 12-15
10829 Berlin
Germany

Email: k.pentikousis@eict.de

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 23, 2016

S. Matsushima
R. Wakikawa
SoftBank

March 22, 2016

Stateless user-plane architecture for virtualized EPC (vEPC)
draft-matsushima-stateless-uplane-vepc-06

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.

- o [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).

- o [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).

- o [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.

```
                        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

In (14), vEPC advertises a routing information of UE to EPC-Es immediately right after the control-plane signaling completion. The routing information contain UE's prefix as destination, remote

endpoint of GTP-U tunnel as next-hop which is S1-U addresses and TEIDs of serving eNodeB/EPC-E, and also QoS class applied to the UE.

In this document, the advertising entity is a BGP speaker so that the BGP speaker is required to indicate those in BGP message. To achieve that, the BGP speaker and EPC-E should be capable of (1) BGP Tunnel Encaps Attribute [I-D.ietf-idr-tunnel-encaps] which specifies the form to encode GTP-U endpoints, and (2) Dissemination of Flow Specification Rule [RFC5575] with IPv6 amendment [I-D.ietf-idr-flow-spec-v6] to indicate applied QoS class.

It is noted that the control-plane needs to expose user-plane information of UEs to BGP speaker. The means of how the control-plane and the BGP speaker deal with that is discussed in Section 3.6.

The EPC-E has a peering with the BGP speaker 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.

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.

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.

```
             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

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 RTR 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.

3.6. Interface between Control-plane and BGP Speaker

In Section 3.2 described, mobility control-plane and BGP speakers within a vEPC need an interface to export user-plane information from the control-plane to the BGP speakers. Perhaps many solutions would be developed proprietarily. However, adopting standardized interface will be much appropriate.

Forwarding Policy Configuration Protocol (FPCP) [I-D.ietf-dmm-fpc-cpdp] has been standardized in IETF for that purpose. That provices client function to the mobility control-plane to export user-plane information, and agent function which enables the BGP speakers to receive the user-plane information when it is implemented into them.

User-plane information contains UE’s IP prefix, GTP-U tunnel endpoints of serving eNodeB/EPC-E and applied QoS class. When those information come into the BGP speaker, the agent renders it into BGP attributes which are UE’s IP prefix, GTP-U tunnel endpoints and QoS class are indicated in NLRI, [I-D.ietf-idr-tunnel-encaps] and [RFC5575] with [I-D.ietf-idr-flow-spec-v6] respectively.

The BGP speaker generates BGP UPDATE messages based on that and then advertises it to EPC-E routers. Figure 10 depicts FPCP enabled vEPC in which mobility control-plane and BGP speaker are interfaced through FPCP client and agent functions.

```
       +-----------------------------------------------------+
       | vEPC                                                |
       |        +-----------------------------------+        |
       |        |       Mobility Control-Plane      |        |
       |        |                                   |        |
       |        | +-----+ +-----+ +-----+ +------+ |        |
       |        | | MME | | SGW | | PGW | | PCRF | |        |
       |        | +-----+-+-----+-+-----+-+------+ |        |
       |        | |   Interface to BGP Speaker    | |        |
       |        | |       (Client Function)       | |        |
       |        +-+--------------^---------------+-+        |
       |                         |                           |
       |                         | FPC Protocol              |
       |                         |                           |
       |        +-+--------------v----------------+-+        |
       |        | |         Agent Function        | |        |
       |        | +-------------------------------+ |        |
       |        |                                   |        |
       |        |            BGP Speaker            |        |
       |        +----------------^------------------+        |
       +------------------------|------------------------+
                                |
                                | BGP Peering
                                | w/ Tunnel encaps & Flow-spec
                                |    attributes
                                |
                  +--------------v----------------+
                  |          EPC-E Routers        |
                  +-------------------------------+
```

FPCP enabled vEPC

Figure 10

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.ietf-idr-flow-spec-v6]
McPherson, D., Raszuk, R., Pithawala, B., Andy, A., and S. Hares, "Dissemination of Flow Specification Rules for IPv6", draft-ietf-idr-flow-spec-v6-07 (work in progress), March 2016.

[I-D.ietf-idr-tunnel-encaps]
Rosen, E., Patel, K., and G. Velde, "The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-encaps-01 (work in progress), December 2015.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009, <http://www.rfc-editor.org/info/rfc5575>.

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-dmm-fpc-cpdp]
Liebsch, M., Matsushima, S., Gundavelli, S., and D. Moses, "Protocol for Forwarding Policy Configuration (FPC) in DMM", draft-ietf-dmm-fpc-cpdp-01 (work in progress), July 2015.

[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.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, DOI 10.17487/RFC3963, January 2005, <http://www.rfc-editor.org/info/rfc3963>.

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

[RFC5555] Soliman, H., Ed., "Mobile IPv6 Support for Dual Stack Hosts and Routers", RFC 5555, DOI 10.17487/RFC5555, June 2009, <http://www.rfc-editor.org/info/rfc5555>.

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

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

[RFC6301] Zhu, Z., Wakikawa, R., and L. Zhang, "A Survey of Mobility Support in the Internet", RFC 6301, DOI 10.17487/RFC6301, July 2011, <http://www.rfc-editor.org/info/rfc6301>.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <http://www.rfc-editor.org/info/rfc6333>.

[RFC6459] Korhonen, J., Ed., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, DOI 10.17487/RFC6459, January 2012, <http://www.rfc-editor.org/info/rfc6459>.

[RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", RFC 6877, DOI 10.17487/RFC6877, April 2013, <http://www.rfc-editor.org/info/rfc6877>.

Authors' Addresses

Satoru Matsushima
SoftBank
1-9-1,Higashi-Shimbashi,Minato-Ku
Tokyo 105-7323
Japan

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

Ryuji Wakikawa
SoftBank
1-9-1,Higashi-Shimbashi,Minato-Ku
Tokyo 105-7323
Japan

Email: ryuji.wakikawa@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

B. Sarikaya

L. Xue

October 30, 2017

Distributed Mobility Management Protocol for WiFi Users in Fixed Network

draft-sarikaya-dmm-for-wifi-05.txt

Abstract

As networks are moving towards flat architectures, a distributed approach is needed to mobility management. This document presents a use case distributed mobility management protocol called Distributed Mobility Management for Wi-Fi. The protocol is based on mobility aware virtualized routing system with software-defined network support. Routing is in Layer 2 in the access network and in Layer 3 in the core network. Smart phones access the network over IEEE 802.11 (Wi-Fi) interface and can move in home, hotspot and enterprise buildings.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Centralized mobility anchoring has several drawbacks such as single point of failure, routing in a non optimal route, overloading of the centralized data anchor point due to the data traffic increase, low scalability of the centralized route and context management [RFC7333].

In this document, we define a routing based distributed mobility management protocol. The protocol assumes a flat network architecture as shown in Figure 1. No client software is assumed at the mobile node.

IP level mobility signaling needs to be used even when MN is connected to a home network or a hotspot. Distributed anchors in the protocol are called Unified Gateways and they represent an evolution from the Broadband Network Gateway (BNG) currently in use.

2. Conventions and Terminology

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 RFC 2119 [RFC2119].

This document uses the terminology defined in [I-D.ietf-dmm-deployment-models] and [I-D.matsushima-stateless-uplane-vepc].

3. Overview

This section presents an overview of the protocol, Distributed Mobility Management for Wi-Fi protocol (DMM4WiFi). See also Figure 1.

Access routers (AR) are Unified Gateways (UGW) that are the access network gateways that behave similarly as Evolved Packet Core (EPC) Edge Router (EPC-E) in [I-D.matsushima-stateless-uplane-vepc]. UGW is configured an anycast address on the interface facing the Residential Gateway (RG). RGs use this address to forward packets from the users. The fixed access network delivers the packets to geographically closest UGW. UGW plays the role of Access Data Plane Node (A-DPN) defined in [I-D.ietf-dmm-deployment-models]. A-DPN and UGW are interchangeably used in this document.

```
       Cloud                        _.---------+----------.
                          ,' ' ---''Virtualized Control Plane---'-.
                         (    +---+          +---+         +---+   '.
                          '.  |VM1|          |VM2|         |VM3|      )
                              +---+          +---+         +--,+    ,'
       IP Routers               |     _.---------+----------.
       with SDN Clients       ,----''           |           '---'-.
       and Agents          ,-'  |               |               \ '-.
                         ,'     |              |                ,   '.
                       (        |    IP Network|                 \     )
                        '.     |               |                  ,   ,'
                          '-. |                ;                  ,\'
                             ;-----. ---------+------------------
                         +-------+         +-------+      +-------+
                         | A-DPN |         | A-DPN |      | A-DPN |
                         +-------+         +-------+      +-------+
                        ,'                     |                    '.
                      (             Access Network                    )
                        '.                   |                    ,'
                         +-----+           +-----+        +-----+
                         | RG  |           | RG  |        | RG  |
                         +-----+           +-----+        +-----+
                      +-----+                           +-----+
                      | MN  | ----move---------------> | MN  |
                      +-----+                           +-----+
```

Figure 1: SDN Based Architecture of Wi-Fi Protocol

Wi-Fi smart phone, the mobile node (MN) is assigned a unique prefix using either Stateless Address Auto Configuration (SLAAC) or by a DHCP server which could be placed in the cloud. In case of SLAAC, RG is delegated the prefixes by DHCP server using [RFC3633].

Prefix assignments to MNs are consistent with the prefixes assigned to UGWs that are shorter than /64. These prefixes are part of the operator's prefix(es) which could be /32, /24, etc.

The mobile node can move at home or in a hot spot from one Access Point (AP) to another AP and MN mobility will be handled in Layer 2 using IEEE 802.11k and 802.11r. Authentication is handled in Layer 2 using [IEEE-802.11i] and [IEEE-802.11-2007] (as described in Section 4.4).

When MN moves from one A-DPN into another A-DPN, IP mobility signaling needs to be introduced. In this document we use Handover Initiate/ Handover Acknowledge (HI/HAck) messages defined in [RFC5949]. Handover Initiate message can be initiated by either previous UGW (predictive handover) or the next UGW (reactive UGW).

In reactive handover, RG establishes a new connection with the next UGW when MN moves to this RG and provides previous UGW address. This will trigger the next UGW to send HI message to the previous UGW. Previous UGW sends HAck messages which establishes a tunnel between previous and next UGWs. Previous UGW sends packets destined to MN to the new UGW which in turn sends them to MN.

Note that the mobility signaling just described is control plane functionality, i.e. between Access-Control Plane Nodes (A-CPN). Control plane in our document is moved to the cloud, thus mobility signaling happens at the cloud, possibly between two virtual machines (VM), A-CPNs.

Upstream packets from MN at the new A-DPN establish the initial routing path when MN first enters the system. This path needs to be updated as MN moves from one A-DPN to another, i.e. MN handover. Since MN keeps the prefix initially assigned, after handover, the new upstream path establishment may establish host routes in the upstream routers. This route is refreshed as long as MN stays under the same A-DPN. Handover signaling and subsequent upstream path establishment is very critical because the downstream packets may need to follow the path that is established for MN.

Software-Defined Networking (SDN) is used in DMM4WiFi in both Layer 2 and Layer 3 routing management. In case of Layer 2 routing, the Open Flow Switch Protocol is used as the south bound interface between the SDN Controller and Layer 2 access network switches. Extensible Messaging and Presence Protocol (XMPP) is used as the north bound interface between the SDN controller and DMM4WiFi application. DMM4WiFi Layer 3 routing is based on SDN controllers manipulating Routing Information Bases (RIB) in a subset of the upstream routers. In this case south bound interface is the NETCONF protocol which is based on the Remote Procedure Call (RPC) protocol and YANG. I2RS architecture is used in this context.

Mobile node generates interface identifier using [RFC7217] in SLAAC. With this method, MN interface identifiers will be different when MN moves from one A-DPN to another A-DPN. MN MAY have different IPv6 addresses due to this method of interface identifier generation.

4. Detailed Protocol Operation

In this section, Layer 2 and Layer 3 mobility procedures are explained.

4.1. Layer 2 Mobility in Access Network

In the access network, RG MAC address acts as an identifier for the MN. Access network switches are controlled by SDN. Controller to Switch interface uses a protocol such as Extensible Messaging and Presence Protocol (XMPP) [RFC6121]. XMPP is based on a general subscribe-publish message bus. SDN controller publishes forwarding instructions to the subscribing switch. Forwarding instructions could be Open Flow like match-forward instructions. Open Flow protocol can also be used [ONFv1.5].

Access network is organized as interconnected switches. The switch connected to the RG is called egress switch. The switch connected to the UGW is called ingress switch. IEEE 802.1ad standard for VLAN (Q-in-Q) is used in the access network, where S-VLAN denotes RG groups, and C-VLAN determines traffic classes. One S-VLAN tag is assigned to create one or more VLAN paths between egress and ingress switches.

MN mobility in the access network can be tracked by keeping a table consisting of MN IP address and RG MAC address pairs. In this document SDN controllers keep the mobility table. This table is used to select proper S-VLAN downstream path from ingress switch to egress switch and upstream path from egress switch to ingress switch.

After a new MN with WiFi associates with RG, RG sends an Unsolicited Neighbor Advertisement (NA) message upstream. This NA message is constructed as per [RFC4861] but the Source Address field is set to a unicast address of MN. NA message is received by SDN controller and it enables SDN controller to update the mobility table. SDN controller selects proper path including S-VLAN and ingress switch to forward the traffic from this MN. The controller establishes the forwarding needed on these switches [IEEE-Paper], i.e. Layer 2 route.

The packet eventually reaches the closest UGW due to the anycast addressing used at the access network interfaces. UGW forwards this packet to the upstream router and so on. The upstream router establishes a route for MN in its routing table with MN’s prefix and with the UGW as the next hop. Prefixes in those routes get smaller and smaller as the packet moves upstream in the routing hierarchy. The routing protocol used could be BGP or other protocols like IS-IS.

4.2. Layer 3 Mobility and Routing in Core Network

MN moving from one RG to another may eventually require MN moving from one A-DPN to another. This is Layer 3 mobility.

Predictive handover happens when MN just before leaving the previous RG (pRG) for the next RG (nRG) MN is able to send an 802.11 message

containing MN MAC address and nRG MAC address, e.g. learned from beacons to the pRG (called Leave Report in Figure 2. pRG then sends a handover indication message to pUGW providing MN and nRG addresses (called Leave Indication) and this could happen between two respective virtual machines in the cloud. This message results in pUGW getting nUGW information and then sending Handover Initiate message to nUGW, which also could happen in the cloud. nUGW replies with Handover Acknowledge message. pUGW sends any packets destined to MN to nUGW after being alerted by the control plane. MN moves to nRG and nUGW is informed about this from Layer 2 mobility Section 4.1. uGW delivers MN’s outstanding packets to MN.

```
          MN         P-RG       N-RG        (P-UGW)     (N-UGW)   Cloud
          |   Leave   |          |            |           |        |
     (a)  |--Report-->|          |            |           |        |
          |           |          |            |           |        |
          |           |       Leave           |           |        |
     (b)  |           |------indication------>|           |        |
          |           |          |            |           |        |
          |           |          |            |           |        |
     (c)  |           |          |            |----HI---->|        |
          |           |          |            |           |        |
          |           |          |            |           |        |
     (d)  |           |          |            |<---HAck---|        |
          |           |          |            |===========|        |
```

Figure 2: Predictive Handover

Reactive handover handover happens when MN attaches the new RG from the previous RG, called Join Report in Figure 3. MN is able to signal in 802.11 association messages previous RG MAC address. nUGW or A-CPN receives new association information together with pRG information, possibly in the cloud (called Handover Indication). nUGW finds pUGW address and sends HI message to pUGW, again happening between two virtual machines in the cloud. pUGW after receiving indication from the cloud server delivers any outstanding MN’s packets to nUGW which in turn delivers them to MN.

```
           MN         P-RG       N-RG        (P-UGW)     (N-UGW)   Cloud
            |   Join    |          |            |           |        |
      (a)   |--Report-------------->|            |           |        |
            |           |          |       Handover         |        |
      (b)   |           |          |------Indication------->|        |
            |           |          |            |           |        |
      (c)   |           |          |            |<----HI----|        |
            |           |          |            |           |        |
      (d)   |           |          |            |----HAck-->|        |
            |           |          |            |           |        |
      (e)   |           |          |            |<--------->|        |
            |           |          |            |           |   data |
      (f)   |           |          |            |===========|        |
```

Figure 3: Reactive Handover

Note that Handover Initiate and Handover Acknowledge messages used in this document carry only a subset of parameters defined in [RFC5949]. Also no involvement with the Local Mobility Anchor (LMA) [RFC5213] is needed.

4.3. Route Establishment

After handover, SDN route establishment in upstream routers needs to take place. In this case NETCONF protocol [RFC6241] and YANG modeling [RFC6020] are used.

Client and Server exchange their capabilities using NETCONF message layer message called hello messages. Client builds and sends an operation defined in YANG module, encoded in XML, within RPC request message [RFC6244]. Server verifies the contents of the request against the YANG module and then performs the requested operation and then sends a response, encoded in XML, in RPC reply message.

Defining configuration data is the primary focus of YANG. Configuration data is writable (rw - read-write) data that is required to transform a system from its initial default state into its current state. There is also state data (ro - read-only) which is a set of data that has been obtained by the system at runtime. An example is routing table changes made by routing protocols in response to the ongoing traffic.

A YANG module for routing management is given in [I-D.ietf-netmod-routing-cfg]. The core routing data model consists of three YANG modules, ietf-routing, ietf-ipv4-unicast-routing, ietf- ipv6-unicast-routing. The core routing data model has two trees: configuration data and state data trees. "routing-instance" or "rib" trees have to

be populated with at least one entry in the device, and additional entries may be configured by a client. Normally the server creates the required item as an entry in state data. Additional entries may be created in the configuration by a client via the NETCONF protocol using RPC messages like edit-config and copy-config.

The user may provide supplemental configuration of system- controlled entries by creating new entries in the configuration with the desired contents. In order to bind these entries with the corresponding entry in the state data list, the key of the configuration entry has to be set to the same value as the key of the state entry.

RPC get message can be used to retrieve all or part of the running configuration data store merged with the device’s state data. RPC get-config operation retrieves configuration data only. RPC fib-route message defined in [RFC8022] retrieves a routing instance for the active route in the Forwarding Information Base (FIB) which is the route that is currently used for sending datagrams to a destination host whose address is passed as an input parameter. So fib-route message plays the role of show route command line interface command.

NETCONF protocol and ietf-routing YANG module can be used for route establishment after handover. As a result for MNs that handover, upstream routing that takes place is not modified up to the lowest level of routers. The lowest level of routers handle the mobility but only proper modifications are needed so that the packets reach the right Unified Gateway, i.e. nUGW.

I2RS Agent as NETCONF Server in nUGW and in pUGW inform the handover to I2RS Clients as NETCONF Client upstream. I2RS Agent at pUGW removes any routing information for MN by first using get-config to retrieve the active route for MN and then an edit-config message with delete operation to delete the active route making sure that the same key is used.

I2RS Agent in nUGW after the handover needs to add a new routing table entry for MN. Due to the topological correctness of MN’s prefix, the new route could be a host route. Next this route is propagated upstream. In this case, nUGW starts the process. SDN Controller as I2RS Client knows that MN handover is successfully completed. SDN Controller starts the upstream route establishment process starting with the I2RS Agent at the upstream router. Either a new route or the host route is added with shorter prefix. Route propagation continues until MN’s prefix becomes topologically correct at which point route propagation stops.

Route propagation at the lowest level starts with I2RS Agent as NETCONF Server in nUGW informing the handover to I2RS Client as NETCONF Client upstream. I2RS Client then checks any routing information for MN by first using get-config to retrieve the active route for MN to make sure that none exits and MN prefix is topologically incorrect. Next I2RS client issues an edit-config message with create operation to add a host route for the new MN. I2RS Client then informs this route to I2RS Client upstream which creates a similar route at the I2RS Agent upstream.

In Appendix A, we present our experimental work using YANG data modelling language which has its own syntax and NETCONF protocol which is XML-based remote procedure call (RPC) mechanism. HTTP based RESTCONF could also be used in a similar way. Two RPC call examples are given. RPC call in Appendix A.3 shows a get-config filter with rtr0 as the key and it is used to retrieve a specific route with a given destination prefix and next hop address. RPC call in Appendix A.4 shows an example edit-config create operation to create a new route with specific route parameters.

4.4. Authentication

Extensible Authentication Protocol (EAP)[RFC3748] is preferred for MN authentication in IEEE 802.11 (Wi-Fi) network. When a MN tries to connect to the WiFi, it needs to mutually authenticate with the network server first. A successful EAP authentication procedure must result in a Pairwise Master Key(PMK) (defined in [IEEE-802.11i]) for the traffic encryption between the MN and the AR.

When a MN moves at home or in a hot spot from one AP to another AP in the same UGW, it is possible that it may to undergo a full EAP authentication (as defined in[RFC3748]). However, there are several simplified authentication methods (defined in [IEEE-802.11-2007]):

o Preauthentication: When The MN supplicant may authenticate with both pRG and nRG at a time. Successful completion of EAP authentication between the MN and nRG establishes a pair of PMKSA on both the MN and nRG. When the MN moves to the nRG, the authentication has already done, which is shown as follows.

```
                              +---------------+
                              | Authentication|
                              |    Server     |
                              +--*----*-------+
          Preauthentication <..........*
                                 /      *
                            +---------*-----+
                            |/        *UGW  |
                            *---------*-----+
                   ,'        /          * |                    '.
                  (         /   Access  *Network                 )
                   '.      /            * |                   ,'
                     +-----*           +-*---+        +-----+
                     | RG /|     ******* RG  |        | RG  |
                     +----/+ *****     +-----+        +-----+
                  +-----****             +-----+
                  | MN  | ----move-----> | MN  |
                  +-----+                +-----+
```

o Cached PMK: The RG reserves the PMK as a result of previous authentication. When the MN is roaming back to the previous RG, if a successful EAP authentication has happened. The MN can retain the 802.11 connection based on PMK information reserved. When the authentication is handled by the UGW as an Authenticator. When the MN moves to the nRG, a join report packet will be initiated from the MN to nRG for IEEE802.11 connection to the same UGW. The nRG can retain the PMK information from the UGW which is reserved during the successful authentication procedure between the MN and the pRG, as shown in Figure 4.

```
                          +---------------+
                          | Authentication|
                          |    Server     |
                          +--*------------+
         Authentication <....*
                           /
                          *-------------+
                         /|    UGW      | PMK Cache
                        / +-------------+   /
                 ,'        /            |      /             '.
                (         /   Access   Network/                )
                 '.      /              |    /               ,'
                  +-----*           +-----+</      +-----+
                  | RG /|           | RG  |        | RG  |
                  +----/+           +-----+        +-----+
                +-----                 +-----+
                | MN  | ----move-----> | MN  |
                +-----+                +-----+
```

Figure 4: Cached PMK-UGW Authenticator

When a MN moves at home or in a hot spot from one AP to another AP in the same UGW, it is possible that it may to undergo a full EAP authentication (as defined in[RFC3748]). However, there are several simple authentication methods (defined in [IEEE-802.11-2007]):

When MN moves from one UGW into another UGW, a join report packet will be initiated from the MN to nRG for IEEE802.11 connection. It is possible that it may to undergo a full EAP authentication (as defined in[RFC3748]). However, because of service performance and continuity requirement, the operators prefer to avoid the full EAP authentication. There are several simplied authentication methods (defined in [IEEE-802.11-2007]):

o Preauthentication: MN supplicant may authenticate with both pRG and nRG at a time. Successful completion of EAP authentication between the MN and nRG establishes a pair of PMKSA on both the MN and nRG. When the MN moves to the nRG, the authentication has already been completed, which is shown as follows.

```
                            +---------------+
                            | Authentication|
                            |    Server     |
                            +--*----*-------+
         Preauthentication <..........*
                             /       *
                  +-------+   /     +-*-----+      +-------+
                  |  UGW  |  /      | *UGW  |      |  UGW  |
                  +-------+ /       +-*-----+      +-------+
                 ,'        /          * |                    '.
                (         /   Access  *Network                 )
                 '.      /            * |                    ,'
                  +-----*           +-*---+        +-----+
                  | RG /|     ******* RG  |        | RG  |
                  +----/+ *****     +-----+        +-----+
                +-----****             +-----+
                | MN  | ----move-----> | MN  |
                +-----+                +-----+
```

o Cached PMK: The RG reserves the PMK as a result of previous authentication. When the MN is roaming back to the previous RG, if a successful EAP authentication has happened. The MN can retain the 802.11 connection based on PMK information reserved. When the authentication is handled by the UGW as an Authenticator. When the MN moves to the nRG, a join report packet will be initiated from the MN to nRG for IEEE802.11 connection to nUGW. The nRG can retain the PMK information from the nUGW, the nUGW may can retain the reserved PMK from the pUGW based on HI message.

```
                          +---------------+
                          | Authentication|
                          |    Server     |
                          +--*------------+
         Authentication <..../
                            /
                   +---------+ HI(PMK Q)+---------+
         PMK Cached|   pUGW  |<-------- |  nUGW   |
                   ++--------+ -------> +--------++ ^  Join Report Msg
                ,'    |   /      HAck(PMK)         |  |     '.
               (      |  /                         |  |        )
                '.    | /    Access   Network      |  |     ,'
                  +----+*           +-----+        ++--+-+
                  | RG /|           | RG  |        | RG  |
                  +----/+           +-----+        +-----+
               +-----                            +-----+
               | MN  | ----move--------------- | MN  |
               +-----+                           +-----+
```

The above Layer 2 operations do not affect Layer 3. MN does not change the prefix assigned to it initially.

Note that charging solution is not described in this version.

5. Multicast Support

Multicast communication to the mobile nodes can be supported with an Multicast Listener Discovery (MLD) Proxy at the Unified Gateway [RFC4605]. Downstream protocol operations between the UGW and the mobile nodes, is the MLD protocol [RFC3810]. Both any source and source specific multicast are supported.

The mobile nodes send MLD Report message when joining a multicast group [RFC3590]. UGW or MLD Proxy sends an aggregated join message upstream. MN and UGW interface works as described in [RFC6224]. After MN joins the group it starts to receive multicast data.

After a handover the mobile node moves to the next UGW, the next UGW needs to get membership or listening state of this MN containing group address and source list. For this purpose, Active Multicast Subscription mobility option (Type 57 for IPv6) [RFC7161] can be used to transfer mobile node’s multicast context or subscription information from the previous UGW to the next UGW, as explained below.

In case of predictive handover, pUGW and nUGW follow the sequence of steps shown in Figure 2. In case MN has multicast context established before handover pUGW MUST transfer MN’s multicast context to nUGW. pUGW MUST add Active Multicast Subscription mobility option to HI message.

For reactive handover pUGW and nUGW follow the sequence of steps shown in Figure 3. In case MN has multicast context established before handover pUGW MUST transfer MN’s multicast context to nUGW. pUGW MUST add Active Multicast Subscription mobility option to HAck messeage.

After receiving the multicast context, nUGW upstream joins any new multicast groups on behalf of MN. Downstream, nUGW maps downstream point-to-point link to a proxy instance.

5.1. IPv4 Support

IPv4 can be supported similarly as in vEPC [I-D.matsushima-stateless-uplane-vepc]. UGW stays as IPv6 node receiving from all RGs IPv6 packets and forwarding them upstream.

IPv4 MN is supported at the RG. RG has B4 functionality of DS-Lite [RFC6333], CLAT entity for 464XLAT [RFC6877], Lightweight B4 [RFC7596] or MAP Customer Edge [RFC7597]. RG encapsulates IPv4 packets using these protocols into IPv6 packets making sure that UGW stays IPv6 only.

6. IANA Considerations

TBD.

7. Security Considerations

This document introduces no extra new security threat. Security considerations stated in [RFC7921] and [I-D.ietf-dmm-deployment-models] apply.

8. Acknowledgements

We would like to thank Ladislav Lhotka, Satoru Matsushima for valuable advice.

9. References

9.1. Normative References

[I-D.ietf-dmm-deployment-models]
Gundavelli, S. and S. Jeon, "DMM Deployment Models and Architectural Considerations", draft-ietf-dmm-deployment-models-02 (work in progress), August 2017.

[IEEE-802.11-2007]
IEEE, "Institute of Electrical and Electronics Engineers, "Telecommunications and information exchange between systems-Local and metropolitan area networks specific requirements -Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", March 2007.

[IEEE-802.11i]
IEEE, "Institute of Electrical and Electronics Engineers, "Unapproved Draft Supplement to Standard for Telecommunications and Information Exchange Between Systems-LAN/MAN Specific Requirements -Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Specification for Enhanced Security,", September 2004.

[ONFv1.5] ONF, "Open Networking Foundation, "OpenFlow Switch Specification Version 1.5.0 (Protocol version 0x06)", January 2015.

[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>.

[RFC3590] Haberman, B., "Source Address Selection for the Multicast Listener Discovery (MLD) Protocol", RFC 3590, DOI 10.17487/RFC3590, September 2003, <https://www.rfc-editor.org/info/rfc3590>.

[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>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <https://www.rfc-editor.org/info/rfc3810>.

[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>.

[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>.

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

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6121] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, DOI 10.17487/RFC6121, March 2011, <https://www.rfc-editor.org/info/rfc6121>.

[RFC6224] Schmidt, T., Waehlisch, M., and S. Krishnan, "Base Deployment for Multicast Listener Support in Proxy Mobile IPv6 (PMIPv6) Domains", RFC 6224, DOI 10.17487/RFC6224, April 2011, <https://www.rfc-editor.org/info/rfc6224>.

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

[RFC6244] Shafer, P., "An Architecture for Network Management Using NETCONF and YANG", RFC 6244, DOI 10.17487/RFC6244, June 2011, <https://www.rfc-editor.org/info/rfc6244>.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <https://www.rfc-editor.org/info/rfc6333>.

[RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", RFC 6877, DOI 10.17487/RFC6877, April 2013, <https://www.rfc-editor.org/info/rfc6877>.

[RFC7161] Contreras, LM., Bernardos, CJ., and I. Soto, "Proxy Mobile IPv6 (PMIPv6) Multicast Handover Optimization by the Subscription Information Acquisition through the LMA (SIAL)", RFC 7161, DOI 10.17487/RFC7161, March 2014, <https://www.rfc-editor.org/info/rfc7161>.

[RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <https://www.rfc-editor.org/info/rfc7217>.

[RFC7596] Cui, Y., Sun, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the Dual-Stack Lite Architecture", RFC 7596, DOI 10.17487/RFC7596, July 2015, <https://www.rfc-editor.org/info/rfc7596>.

[RFC7597] Troan, O., Ed., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, Ed., "Mapping of Address and Port with Encapsulation (MAP-E)", RFC 7597, DOI 10.17487/RFC7597, July 2015, <https://www.rfc-editor.org/info/rfc7597>.

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <https://www.rfc-editor.org/info/rfc8022>.

9.2. Informative References

[I-D.matsushima-stateless-uplane-vepc]
Matsushima, S. and R. Wakikawa, "Stateless user-plane architecture for virtualized EPC (vEPC)", draft-matsushima-stateless-uplane-vepc-06 (work in progress), March 2016.

[IEEE-Paper]
"Jyotirmoy Banik, et al., "IEEE 24th International Conference on Computer Communication and Network 2015, "Enabling Distributed Mobility Management: A Unified Wireless Network Architecture Based on Virtualized Core Network", DOI: 10.1109/ICCCN.2015.7288404",", August 2015.

[RFC7333] Chan, H., Ed., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", RFC 7333, DOI 10.17487/RFC7333, August 2014, <https://www.rfc-editor.org/info/rfc7333>.

[RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <https://www.rfc-editor.org/info/rfc7921>.

Appendix A. YANG and RPC Programs

In this annex, we present our YANG and RPC solutions.

A.1. Host Routing Module

We first obtained host routing YANG module using IPv6 unicast routing module (ietf-ipv6-unicast-routing) which is part of ietf-routing module. This module defines a list of host routes which contain host address/prefix and corresponding next hop address.

A.2. Route Establishment RPCs

This program runs on ietf-ipv6-unicast-host-routing YANG module which has been obtained from ietf-ipv6-unicast-routing module by defining the hostroute as a list of host routes. First issue a get-config on the configuration data to extract the existing route for the host whose prefix is destination-prefix and the next-hop is the next-hop address. Delete the route at pUGW. This procedure deletes the route at pUGW.

```
<rpc message-id="101" ... >

get-config(running, filter=(destination-prefix, next-hop-address))

// check the reply, make sure it is OK, i.e. does not contain <rpc-
error> element.

edit-config(running, delete, config)
```

Add a new route for MN at nUGW. This route is based on MN's prefix, destination-prefix and the upstream router to which MN's traffic should routed, next-hop-address.

```
<rpc message-id="101" ... >

get-config(running, filter=(destination-prefix, next-hop-address))

// check the reply, make sure it is an error, i.e. it contains <rpc-
error> element of type application and tag data-missing i.e. no route
exists

edit-config(running, create, config)
```

Add a new host route for MN at nUGW. This route is added in case MN's prefix is not topologically correct at nUGW and routers above.

```
<rpc message-id="101" ... >

 get-config(running, filter=(destination-prefix, next-hop-address))

// check the reply, make sure it is an error, i.e. it contains <rpc-
error> element of type application and tag data-missing, i.e. no
route exists

edit-config(running, create, config)
```

We next show in Appendix A.3 and Appendix A.4 example RPC procedures for get-config and edit-config. Some arbitrary values for destination prefix and next hop address are used.

A.3. get-config RPC procedure for host routes

This RPC procedure shows a get-config filter to find a record in the routing information base for a specific host whose prefix is 2001:db8:1:0::/64 and the next-hop is 2001:db8:0:1::2. It could be used for the get-config's in Appendix A.2. We validated this procedure using the public domain tool pyang.

```
      <rpc message-id="101"
               xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
      xmlns:v6ur="urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing"
      xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
      xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
      xmlns:ip="urn:ietf:params:xml:ns:yang:ietf-ip"
      xmlns:rt="urn:ietf:params:xml:ns:yang:ietf-routing">
       <get-config>
         <source>
           <running/>
         </source>
         <filter type="subtree">
          <t:top xmlns:t="urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-host-rou
ting">
         <t:routing-instance> rtr0 </t:routing-instance>


           <t:rib>
           <t:routes>
           <t:route>
             <t:destination-prefix>
            2001:db8:1:0::/64
           </t:destination-prefix>
             <t:outgoing-interface>eth1</t:outgoing-interface>
           <t:next-hop-address>
                         2001:db8:0:1::2
             </t:next-hop-address>
             </t:route>
           </t:routes>
         </t:rib>
       </t:top>
     </filter>
       </get-config>
     </rpc>
```

A.4. edit-config RPC procedure to create a host route

This RPC procedure shows an edit-config procedure to create a new host route in the routing information base for a specific host whose prefix is 2001:db8:1:0::/64 and the next-hop is 2001:db8:0:1::2. It could be used for the edit-config’s in Appendix A.2. We validated this procedure using the public domain tool pyang.

```
         <rpc message-id="101"
               xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
      xmlns:v6ur="urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing"
      xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
      xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
      xmlns:ip="urn:ietf:params:xml:ns:yang:ietf-ip"
      xmlns:rt="urn:ietf:params:xml:ns:yang:ietf-routing">
       <edit-config>
         <target>
           <running/>
         </target>
         <default-operation>none</default-operation>
         <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
          <top xmlns="urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-host-routing
">
           <routing-instance> rtr0 </routing-instance>
           <rib>
           <routes>
            <route xc:operation="create">
             <destination-prefix >
            2001:db8:1:0::/64
             </destination-prefix>
             <outgoing-interface>eth1</outgoing-interface>
             <next-hop-address>
                         2001:db8:0:1::2
              </next-hop-address>
             </route>
           </routes>
         </rib>
       </top>
     </config>
    </edit-config>
   </rpc>
```

Authors' Addresses

Behcet Sarikaya

Email: sarikaya@ieee.org

Li

Email: xueliucb@gmail.com

DMM WG
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2015

P. Seite
Orange

July 3, 2014

Multihoming support for Residential Gateway (RG) using IP mobility protocols
draft-seite-dmm-bonding-00.txt

Abstract

The Quality of Experience of fixed network user can be improved with multiple WAN interfaces Residential Gateway (RG), i.e. RG supporting more than one WAN interface (e.g. LTE and DSL), so that it can take benefit of multihoming advantages. This document discusses the use of IP mobility protocols (NEMO [RFC3753] and Mobile IPv6 [RFC6275]), and their Multiple care-of-address extension [RFC5648], to meet multihomed RG requirements. This document also defines a new mobility option, the bonding option, for IP mobility protocols. This option is used by the mobility entities to configure the interface bonding where packets, of a given IP flow, are distributed.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Fix access network (e.g. DSL) usually provides Internet connectivity via a Residential Gateway (RG) acting as the access router. When equipped with different WAN access technologies (e.g. DSL and LTE), the RG could take benefit of multihoming advantages such as redundancy, load sharing, load balancing and so on. Among multihoming benefits is the bandwidth aggregation, so that increased bandwidth is provided to the end-user by allowing the RG to use simultaneously the available WAN interfaces. The RG can either bind some given IP flows to given interfaces or distribute the uplink packets of a same IP flow to more than one WAN interface (i.e. interface bonding). On the network side, an aggregation gateway performs same traffic management operations for downlink traffic.

Actually, the architecture described above is typically a mobile network architecture; functionally, the aggregation point is nothing else than an IP mobility anchor and the RG can be viewed as a mobile router. Actually, if IP mobility protocols have been specified to bring IP session continuity for mobile hosts or mobile networks, nothing prevent to use them in a fixed network context [RFC4908]. Besides, IP mobility protocols can meet a basic aggregation requirement, which consist in setting-up dynamically forwarding paths, over more than one access network, between the RG and a traffic anchoring in charge of managing bandwidth aggregation. Typically, Mobile IPv6 [RFC6275]), NEMO [RFC3963]) and MCoA [RFC5648]) can be used in bandwidth aggregation context to establish forwarding paths (i.e. bindings) on a Residential Gateway with more than one WAN access (e.g. xDSL and LTE, connection to several xDSL ISPs). This document briefly discusses these architectures on Section 3.

IP mobility protocols can be used without any modifications to bring bandwidth aggregation at the IP flow level: a multihomed RG can use simultaneously all its WAN interfaces and binds different IP flows to different interfaces. However, for bandwidth aggregation at the packet level, the way to use the available mobility bindings may differ from legacy IP mobility solutions. Indeed, IP mobility protocols tend to associate a given IP flow to a given binding ([RFC6089]), while interface bonding use-case may require to distribute an IP flow simultaneously over more than one binding, i.e. perform bonding of the WAN interfaces for higher bandwidth. This document specifies IP mobility extensions to allow this behaviour.

2. Conventions and Terminology

2.1. 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 RFC 2119 [RFC2119].

2.2. Terminology

All the mobility related terms used in this document are to be interpreted as defined in the Mobile IPv6 specifications [RFC3753], [RFC5648], [RFC5213] and [RFC6275].

3. Architecture

This section proposes to use a NEMO [RFC3963] architecture in a fix network context to allow aggregation of the WAN interfaces of an Residential Gateway (RG).

The Residential Gateway has more than one WAN interfaces (e.g. DSL and LTE), from which it obtains IP addresses, i.e. care-of-address, via legacy IP allocation mechanisms (e.g. DHCP, SLAAC). Then, the RG registers these care-of-addresses to the mobility anchor using NEMO [RFC3963]) protocol and multiple care-of-addresses [RFC5648] extension. Mobile IP bi-directional tunnels are established, between the RG and the mobility anchor, over each WAN interface. The RG has a unique Home Address through which it is reachable when it is registered with its Home Agent. The Home Address is configured from a prefix advertised by its Home Agent. When the Home Agent receives a data packet meant for a node in the RG Network, it tunnels the packet to the RG to one of the available care-of address. The selection of the care-of-address depends on the aggregation method, operating either at the IP flow or at the packet level:

- At the IP flow level: this scenario is just an application of the current IP flow mobility solution [RFC6089]). The home agent forwards the packets according IP flow routing rules, which give association between IP flows and bindings, received from the RG. The latter indicates flow routing rules to the home agent using flow binding extensions for NEMO ([RFC6088] and [RFC6089]).

- At the Packet Level: in this scenario, IP flow management slightly differs from the default mobile IP behaviour; the home agent distributes packets, belonging to a same IP flow, over more than one bindings simultaneously. The home agent should select the bindings according to interface aggregation indication provided by the RG with the bonding option described in Section 4.1. Note that specification of packets distribution schemes is out the scope of this document.

When receiving a packet, the RG decapsulates the packet and forwards it onto the RG Network. If aggregation operates at the packet level, the RG may buffer and reorder packets before delivery. Buffering and reordering considerations are out of the scope of this document.

```
                      IP Network #1                     HA Binding Cache
 +------------+        _--------_    +------------+   ==================
 |            | BID#1 (          )   |            |   RG, BID#1[HoA, CoA#1],BID#
2[HoA, CoA#2]
 |Residential +======(==IP-in-IP==)==+            |
 | Gateway    |       (_        _)   |Aggregation |
 |  (RG)      |         (_______)    | Gateway    |
 |            |                      |(Home Agent |------>
 |  Mobility  |                      |            |
 |  Client    |        IP Network #2 |            |
 |            |        _--------_    |            |
 |            | BID#2 (          )   |            |
 |            +======(==IP-in-IP==)==+            |
 |            |       (_        _)   |            |
 +-----+------+         (_______)    +------------+
       |
----RG network----
       |
    end-nodes
```

Figure 1: Multihomed RG architecture

Figure 2: Multihomed MN architecture

4. Protocol Messaging Extensions

4.1. Bonding Option

The Bonding option is a mobility header option used by the mobile client and the home agent to indicate bindings to be aggregated. The option can be used by any IP mobility protocols supporting Multiple care-of-address registration, it is carried within the Binding Update, Binding Acknowledgement, UPN/UPA and Binding Refresh Request.

The alignment requirement for this option is 4n.

```
 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      |    BO-ID      |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Bonding Attributes (optional)               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Bonding Option

Type

To be assigned by IANA

Length

8-bit unsigned integer indicating the length of the option in octets, excluding the Type and Length fields.

Bonding Identifier (BO-ID)

The mobile client assigns a BO-ID to each bonding, aggregating at least two bindings. The BO-ID MUST be unique for a given home address. The value is an integer between 0 and 65535. When the value is (0), all . If a mobile node has only one bonding, the assignment of a BO-ID is not needed.

Reserved

This field is unused for now. The value MUST be initialized to a value of (0) by the sender and MUST be ignored by the receiver.

Bonding Attributes

One or more Type-Length-Value (TLV) encoded bonding indication. Attributes are optional.

4.2. Bonding attributes

4.2.1. Bindings list

MUST be included if bonding is expected to apply on a sunset of available bindings. The list of binding IDs indicates at least two bindings that are grouped together within a single BO-ID.

```
 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    |            Reserved           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |              Binding#1        |            Binding#2          ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Bonding Option

The BID is as defined in [RFC5648], it is a 16-bit unsigned integer.

4.2.2. Traffic Selector

MUST be included if only some given IP flows are expected to take benefit of the interfaces bonding.

```
  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    |   Reserved    |    TS Format  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                       Traffic Selector ...                    ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Bonding Option

Type

2

Length

The length of following data value in octets.

TS Format

An 8-bit unsigned integer indicating the Traffic Selector Format. Value "0" is reserved and MUST NOT be used. The value of (1) is assigned for IPv4 Binary Traffic Selector, as defined in section 3.1 of [RFC6088].

Traffic Selector

variable-length opaque field for including the traffic specification identified by the TS format field.

5. Protocol Considerations

5.1. Sending Bonding Request

The mobile client sends a Binging update with bindings registration and bonding indication to the mobility anchor.

```
 IPv6 header (src=Care-of Address, dst=Home Agent Address)
        IPv6 Home Address Option
        Mobility header
            Binding Update
               Mobility Options
                    Bonding Option
                       Bonding attributes
```

Figure 6: Binding Update with Bonding Request

5.2. Receiving Bonding request

The mobility anchor registers multiple care-of-addresses as per [RFC6088]. If the binding update contains a bonding option while the mobility anchor is not able to the meet the request, the later shall returns a binding acknowledgement without bonding option. If the mobility anchor has bonding capabilities, it shall process the bonding option as follows:

- o The bonding option has no attributes: the mobility anchor configure a forwarding interface bonding all bindings registered for the RG/MN home address/prefix. All the IP traffic sent to the home address will be distributed over this interface.

- o The bonding option carries only Traffic Selector: the mobility anchor configure a forwarding interface bonding all bindings registered for the RG/MN home address/prefix; only packets corresponding to the traffic selectors shall be distributed over this interface.

- o The bonding option carries only list of bindings: the mobility anchor configure a forwarding interface bonding indicated bindings; then, all the IP traffic sent to the home address will be distributed over this interface.

- o The bonding option carries both list of bindings and traffic selector: the mobility anchor configure a forwarding interface bonding indicated bindings; only packets corresponding to the traffic selectors shall be distributed over this interface.

The way the mobility anchor distribute downlink packets on interfaces is out of scope of this document. Note, that it is not mandatory, for the mobility anchor, to use the same distribution scheme than applied at the mobile client side (i.e. RG or MN).

5.3. Tunnelling and packet distribution scheme

By default IP-in-IP tunnelling is used between RG and mobility anchor. However, RG and mobility anchor can negotiate using GRE with GRE Key and sequence number extensions [RFC6088], which, for example, could be used by the recipient to reorder packets before delivery. Methods to buffer and reorder packets is out of the scope of this document.

How to distribute packets on interfaces is out of scope of this document. Proprietary distribution scheme may require mobility entity to share information (e.g. RG sends its DSL synchronisation rate); in this case the Vendor specific mobility option [RFC5094] can be used for that purpose. Mobility entities are not requested to use the same packet distribution scheme.

5.4. Network controlled aggregation

The mobility anchor n enforce its decision to the RG. UPN/UPA could be used to allow the mobility anchor to enforce aggregation rules to the RG. Rules can be either IP flow routing policies or bonding configuration.

6. IANA Considerations

This document requires the following IANA action:

This specification defines a new mobility option, the Bonding option. The format of this option is described in Section 4.1. The type value for this mobility option needs to be allocated from the Mobility Options registry at <http://www.iana.org/assignments/mobility-parameters>.

7. Security Considerations

The Bonding option defined in this specification is for use in Binding Update and Binding Acknowledgement messages. This option is carried in these messages like any other mobility header option. [RFC3963] and [RFC6275] identify the security considerations for these signaling messages. When included in these signaling messages, the Bonding option does not require additional security considerations.

8. Acknowledgements

The author would like to thank Sri Gundavelli and Gaetan Feige for having shared thoughts on concepts exposed in this document.

9. References

9.1. Normative References

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

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC3753] Manner, J. and M. Kojo, "Mobility Related Terminology", RFC 3753, June 2004.

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

[RFC5094] Devarapalli, V., Patel, A., and K. Leung, "Mobile IPv6 Vendor Specific Option", RFC 5094, December 2007.

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

[RFC5648] Wakikawa, R., Devarapalli, V., Tsirtsis, G., Ernst, T., and K. Nagami, "Multiple Care-of Addresses Registration", RFC 5648, October 2009.

[RFC6088] Tsirtsis, G., Giarreta, G., Soliman, H., and N. Montavont, "Traffic Selectors for Flow Bindings", RFC 6088, January 2011.

[RFC6089] Tsirtsis, G., Soliman, H., Montavont, N., Giaretta, G., and K. Kuladinithi, "Flow Bindings in Mobile IPv6 and Network Mobility (NEMO) Basic Support", RFC 6089, January 2011.

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

9.2. Informative References

[RFC4908] Nagami, K., Uda, S., Ogashiwa, N., Esaki, H., Wakikawa, R., and H. Ohnishi, "Multi-homing for small scale fixed network Using Mobile IP and NEMO", RFC 4908, June 2007.

Author’s Address

Pierrick Seite
Orange
4, rue du Clos Courtel, BP 91226
Cesson-Sevigne 35512
France

Email: pierrick.seite@orange.com

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:

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

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

- o 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.

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

- o 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:

- o 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.

- o 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
Intended status: Standards Track
Expires: September 8, 2015

A. Yegin
Unaffiliated
K. Kweon
J. Lee
J. Park
Samsung
D. Moses
Intel

March 7, 2015

On Demand Mobility Management
draft-yegin-dmm-ondemand-mobility-03

Abstract

Applications differ with respect to whether they need IP session continuity and/or IP address reachability. The network providing the same type of service to any mobile host and any application running on the host yields inefficiencies. This document describes a solution for taking the application needs into account in selectively providing IP session continuity and IP address reachability on a per-socket basis.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 8, 2015.

Table of Contents

1. Introduction

In the context of Mobile IP [RFC5563][RFC6275][RFC5213][RFC5944], following two attributes are defined for the IP service provided to the mobile hosts:

IP session continuity: The ability to maintain an ongoing IP session by keeping the same local end-point IP address throughout the session despite the mobile host chaging its point of attachment within the IP network topology. The IP address of the host may change between two independent IP sessions, but that does not jeopardize the IP session continuity. IP session continuity is essential for mobile hosts to maintain ongoing flows without any interruption.

IP address reachability: The ability to maintain the same IP address for an extended period of time. The IP address stays the same across independent IP sessions, and even in the absence of any IP session. The IP address may be published in a long-term registry (e.g., DNS), and it is made available for serving incoming (e.g., TCP)

connections. IP address reachability is essential for mobile hosts to use specific/published IP addresses.

Mobile IP is designed to provide both IP session continuity and IP address reachability to mobile hosts. Architectures utilizing these protocols (e.g., 3GPP, 3GPP2, WIMAX) ensure that any mobile host attached to the compliant networks can enjoy these benefits. Any application running on these mobile hosts is subjected to the same treatment with respect to the IP session continuity and IP address reachability.

It should be noted that in reality not every application may need those benefits. IP address reachability is required for applications running as servers (e.g., a web server running on the mobile host). But, a typical client application (e.g., web browser) does not necessarily require IP address reachability. Similarly, IP session continuity is not required for all types of applications either. Applications performing brief communication (e.g., DNS client) can survive without having IP session continuity support.

Achieving IP session continuity and IP address reachability by using Mobile IP incurs some cost. Mobile IP protocol forces the mobile host’s IP traffic to traverse a centrally-located router (Home Agent, HA), which incurs additional transmission latency and use of additional network resources, adds to the network CAPEX and OPEX, and decreases the reliability of the network due to the introduction of a single point of failure [I-D.ietf-dmm-requirements]. Therefore, IP session continuity and IP address reachability should be be provided only when needed.

Furthermore, when an application needs session continuity, it may be able to satisfy that need by using a solution above the IP layer, such as MPTCP [RFC6824], SIP mobility [RFC3261], or an application-layer mobility solution. Those higher-layer solutions are not subject to the same issues that arise with the use of Mobile IP since they can utilize the most direct data path between the end-points. But, if Mobile IP is being applied to the mobile host, those higher-layer protocols are rendered useless because their operation is inhibited by the Mobile IP. Since Mobile IP ensures the IP address of the mobile host remains fixed (despite the location and movement of the mobile host), the higher-layer protocols never detect the IP-layer change and never engage in mobility management.

This document proposes a solution for the applications running on the mobile host to indicate whether they need IP session continuity or IP address reachability. The network protocol stack on the mobile host, in conjunction with the network infrastructure, would provide the required type of IP service. It is for the benefit of both the users

and the network operators not to engage an extra level of service unless it is absolutely necessary. So it is expected that applications and networks compliant with this specification would utilize this solution to use network resources more efficiently.

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. Solution

3.1. Types of IP Addresses

Three types of IP addresses are defined with respect to the mobility management.

- Fixed IP Address

This is what standard Mobile IP provides with a Home Address (HoA). The mobile host is configures a HoA from a centrally-located Home Network. Both IP session continuity and IP address reachability are provided to the mobile host with the help of a router in the Home Network (Home Agent, HA). This router acts as an anchor for the IP address of the mobile host.

- Sustained IP Address

This type of IP address provides IP session continuity but not IP address reachability. It is achieved by ensuring that the IP address used at the beginning of the session remains usable despite the movement of the mobile host. The IP address may change after the termination of the IP session(s), therefore it does not exhibit persistence.

A sustained IP address may be configured and maintained by using access network anchoring, corresponding network anchoring, or some other solution.

- Nomadic IP Address

This type of IP address provides neither IP session continuity nor IP address reachability. The IP address is obtained from the serving IP gateway and it is not maintained across gateway changes. In other words, the IP address may be released and replaced by a new IP address when the IP gateway changes due to the movement of the mobile host.

Applications running as servers at a published IP address require a Fixed IP Address. Long-standing applications (e.g., an SSH session) may also require this type of address. Those applications could use a Sustained IP Address, but that can produce sub-optimal results if the mobile host ends up far from the anchor gateway. Enterprise applications that connect to an enterprise network via virtual LAN require a Fixed IP Address.

Applications with short-lived transient IP sessions can use Sustained IP Addresses. For example: Web browsers.

Applications with very short IP sessions, such as DNS client and instant messengers, can utilize Nomadic IP Addresses. Even though they could very well use a Fixed of Sustained IP Addresses, the transmission latency would be minimized when a Nomadic IP Address is used.

3.2. Granularity of Selection

The IP address type selection is made on a per-socket granularity. Different parts of the same application may have different needs. For example, control-plane of an application may require a Fixed IP Address in order to stay reachable, whereas data-plane of the same application may be satisfied with a Sustained IP Address.

3.3. On Demand Nature

At any point in time, a mobile host may have a combination of IP addresses configured. Zero or more Nomadic, zero or more Sustained, and zero or more Fixed IP addresses may be configured on the IP stack of the host. The combination may be as a result of the host policy, application demand, or a mix of the two.

When the application requires a specific type of IP address and such an IP address is not already configured on the host, then the IP stack shall attempt to configure one. For example, a host may not always have a Fixed IP address available as such an address is rarely used. In case an application requests one, then the IP stack shall make an attempt to configure one using Mobile IP. If Mobile IP protocol is not available on the stack, or if its operation fails, then the IP stack shall fail the associated socket request. In case of successful Mobile IP operation, a Fixed IP Address gets configured on the mobile host. If another socket requests a Fixed IP address at a later time, then the same IP address may be served to that socket as well. When the last socket using the requested IP address is closed, the IP address may be released or kept for future applications that may be launched and require a Fixed IP address.

The following are matters of policy, which may be dictated by the host itself, the network operator, or the system architecture standard:

- The initial set of IP addresses configured on the host at the boot time.

- Permission to grant various types of IP addresses to a requesting application.

- Determination of a default address type when an application does not make any explicit indication, whether it already supports the required API or it is just a legacy application.

3.4. Conveying the Selection

The selection of the address type is conveyed from the applications to the IP stack in a way to influence the source address selection algorithm [RFC6724].

The current source address selection algorithm operates on the available set of IP addresses when selecting an address. According to the proposed solution, if the requested type IP address is not available at the time of the request, then the IP stack shall make an attempt to configure one such IP address. The selected IP address shall be compliant with the requested IP address type, whether it is selected among available addresses or dynamically configured. In the absence of a matching type (because it is not available and not configurable on demand), the source address selection algorithm shall return an empty set.

A Socket API-based interface for enabling applications to influence the source address selection algorithm is described in [RFC5014]. That specification defines IPV6_ADDR_PREFERENCES option at the IPPROTO_IPV6 level. That option can be used with setsockopt() and getsockopt() calls to set and get address selection preferences.

Furthermore, that RFC also specifies two flags that relate to IP mobility management: IPV6_PREFER_SRC_HOME and IPV6_PREFER_SRC_COA. These flags are used for influencing the source address selection to prefer either a Home Address or a Care-of Address.

Unfortunately, these flags do not satisfy the aforementioned needs due to the following reasons, therefore new flags are proposed in this document:

- Current flags indicate a "preference" whereas there is a need for indicating "requirement". Source address selection algorithm does

not have to produce an IP address compliant with the "preference" , but it has to produce an IP address compliant with the "requirement".

- Current flags influence the selection made among available IP addresses. The new flags force the IP stack to configure a compliant IP address if none is available at the time of the request.

- The Home vs. Care-of Address distinction is not sufficient to capture the three different types of IP addresses described in Section 2.1.

The following new flags are defined in this document and they shall be used with Socket API in compliance with the [RFC5014]:

IPV6_REQ_FIXED_IP /* Require a Fixed IP address as source */

IPV6_REQ_SUSTAINED_IP /* Require a Sustained IP addr. as source */

IPV6_REQ_NOMADIC_IP /* Require a Nomadic IP address as source */

More than one of these flags may be set on the same socket. In that case, an IP address compliant with any one of them shall be selected. TBD: Disallow this case?

When any of these new flags is used, then the IPV6_PREFER_SRC_HOME and IPV6_PREFER_SRC_COA flags, if used, shall be ignored.

These new flags are used with setsockopt()/getsockopt(), getaddrinfo(), and inet6_is_srcaddr() functions [RFC5014]. Similar with the setsockopt()/getsockopt() calls, getaddrinfo() call shall also trigger configuration of the required type IP address, if one is not already available. When the new flags are used with getaddrinfo() and the triggered configuration fails, the getaddrinfo() call shall ignore that failure (i.e., not return an error code to indicate that failure). Only the setsockopt() shall return an error when configuration of the requested type IP address fails.

Application of this solution to IPv4 is TBD.

4. Backwards Compatibility Considerations

Backwards compatibility support is required by the following 3 types of entities:

- The Applications on the mobile host

- The IP stack in the mobile host

- The network infrastructure

4.1. Applications

Legacy applications that do not support the new flags will use the legacy API to the IP stack and will not enjoy On-Demand Mobility feature.

Applications using the new flags must be aware that they may be executed in environments that do not support On-Demand Mobility feature. Such environments may include legacy IP stack in the mobile host, legacy network infrastructure, or both. In either case, the API will return an error code and the invoking applications must respond with using legacy calls without On-Demand Mobility feature.

4.2. IP Stack in the Mobile Host

New IP stacks must continue to support all legacy operations. If an application does not use On-Demand Mobility feature, the IP stack must respond in a legacy manner.

If the network infrastructure supports On-Demand Mobility feature, the IP stack may still request specific types of source IP address transparently to legacy applications. This may be useful for environments in which both legacy and new applications are executed.

The definition of what type of addresses to request and how they are assigned to legacy applications are outside of the scope of this specification.

4.3. Network Infrastructure

The network infrastructure may or may not support the On-Demand Mobility feature. How the IP stack on the host and the network infrastructure behave in case of a compatibility issue is outside the scope of this API specification.

5. Security Considerations

The setting of certain IP address type on a given socket may be restricted to privileged applications. For example, a Fixed IP Address may be provided as a premium service and only certain applications may be allowed to use them. Setting and enforcement of such privileges are outside the scope of this document.

6. IANA Considerations

TBD

7. Acknowledgements

We would like to thank Alexandru Petrescu, John Kaippallimalil, Jouni Korhonen, Seil Jeon, and Sri Gundavelli for their valuable comments and suggestions on this work.

8. References

8.1. 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.

8.2. Informative References

[I-D.ietf-dmm-requirements] Chan, A., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", draft-ietf-dmm-requirements-17 (work in progress), June 2014.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

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

[RFC5563] Leung, K., Dommety, G., Yegani, P., and K. Chowdhury, "WiMAX Forum / 3GPP2 Proxy Mobile IPv4", RFC 5563, February 2010.

[RFC5944] Perkins, C., "IP Mobility Support for IPv4, Revised", RFC 5944, November 2010.

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

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, January 2013.

Authors' Addresses

Alper Yegin
Unaffiliated
Istanbul
Turkey

Email: alper.yegin@yegin.org

Kisuk Kweon
Samsung
Suwon
South Korea

Email: kisuk.kweon@samsung.com

Jinsung Lee
Samsung
Suwon
South Korea

Email: js81.lee@samsung.com

Jungshin Park
Samsung
Suwon
South Korea

Email: shin02.park@samsung.com

Danny Moses
Intel Corporation
Petah Tikva
Israel

Email: danny.moses@intel.com

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2015

A. Yegin
J. Park
K. Kweon
J. Lee
Samsung

July 3, 2014

IP Mobility Orchestrator
draft-yegin-ip-mobility-orchestrator-00

Abstract

Host stacks can support mobility at multiple layers. Mobility protocols operating at different layers constitute alternate solutions with various pros and cons, and they can also have adverse affects on each other when used simultaneously. Optimal results in terms of seamless handover and data-path optimization can be achieved when execution of these protocols are coordinated.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Host stacks can support mobility at multiple layers, such as network, transport, and application layers. Mobility protocols operating at different layers have different characteristics in terms of availability, support for seamless handovers, and data-path efficiency. No single solution supports both seamless handovers and optimum data-paths while being universally available to all hosts and networks. Furthermore, mobility protocols at different layers can have adverse affect on each other when operating simultaneously (e.g., one blocking the other).

This document describes the problem in detail, and proposes a solution to achieve optimal results by coordinating the execution of multiple mobility protocols.

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. Problem Statement

A number of protocol solutions are available to mobile hosts for maintaining their end-to-end communication sessions while changing their point of attachment within the IP network topology. Such solutions include but are not limited to Mobile IP [RFC6275] [RFC5944], Proxy Mobile IP [RFC5213] [RFC5563], GTP [GTP], LISP [RFC6830], MOBIKE [RFC4555], MPTCP [RFC6824], SCTP [RFC4960], SIP [RFC3261], and the proprietary ones built into the individual applications (such as Instant Messengers). While any of these protocols can maintain session continuity, they have different characteristics.

The solutions that can completely hide IP mobility from the mobile host stack include protocols like Proxy Mobile IP and GTP. These solutions appear to operate below Layer 3 from the mobile host’s stack perspective (hence we call them "sub-IP solutions"). Sub-IP solutions are available to all 3G/4G terminals. Every application on a host attached to such a network can benefit from the mobility service provided by these protocols. These protocols can achieve seamless handovers, thanks to their ability to build data-path extensions between source and target access networks during handovers. Data-path extension can be setup fast because they require short-haul signaling between the nearby access networks. Even though the handovers are seamless, the end-to-end data-paths between the mobile hosts and their corresponding hosts are sub-optimal due to triangular routing via off-path IP anchors.

Protocol solutions operating at IP layer include Mobile IP and MOBIKE. These solutions are not available on all mobile host stacks. When they are available, they can be utilized by any of the applications running on the mobile host. Seamless handover capability and data-path suboptimality handicap apply to this group of solutions for the same reasons as outlined for the sub-IP solutions.

Solutions operating above the IP layer include MPTCP, SCTP, SIP, and application-specific ones. Availability of these protocols cannot be guaranteed on every host. Furthermore, even when they are available, their applicability to applications is limited. For example, MPTCP only applies to TCP-based applications, not to UDP-based applications. Seamless handovers are not possible with these solutions as any handover-related state update requires a long-haul end-to-end signaling with the corresponding host. The round-trip time required for this signaling becomes the source of packet loss and delay during handovers. Inbound packets that are in-flight during the handover procedure are lost, and outbound packets cannot be transmitted until the handover is completed. On the other hand,

the end-to-end data-path is always optimal as the IP packets use topological IP addresses and they are not forced to traverse off-path IP anchors.

Each of these mobility protocols, when present, operate in isolation. They are not aware of each other's presence or state, and they do not coordinate their state machines among each other either. Furthermore, solutions operating at the lower layers negatively impact the solutions operating at the higher layers. For example, MPTCP cannot detect IP subnet change when the host also uses Mobile IP. Mobile IP hides any IP address change from higher-layers, not only from the applications (an intended benefit) but also from the MPTCP implementation (an undesirable side effect). Therefore, a mobile host stack implementing both Mobile IP and MPTCP cannot enjoy the mobility benefits of MPTCP due to Mobile IP operation. This creates a sub-optimal result.

Each solution type has its pros and cons, and there is no clear winner among them. No single solution can provide both seamless handovers and optimal data-paths by itself. Furthermore, solutions can have negative side-effects on each other to the extent that some are rendered useless.

4. Solution

4.1. Approach

Sub-IP and IP-layer solutions can provide seamless handovers but lack data-path optimization. On the other hand, above-IP solutions provide data-path optimization but fail to provide seamless handovers. The ideal solution would be based on coordianted execution of the two types of solutions.

Let's illustrate the solution concept in action on a simple call flow. Consider the case where both the mobile host and its corresponding host support MPTCP, and the access network supports Proxy Mobile IP.

```
     Mobile                                                Corresponding
      Host                          t-GW       s-GW        Host
       |                             |          |           |
       |<---1. configure IP1------------------>|           |
       |                             |          |           |
       |<...2. start e2e IP flow .............o.........>|
       |                             |          |           |
       * 3. attach to t-GW           |          |           |
       |                             |          |           |
       |<---4a. configure IP2------>|           |           |
       |                             |          |           |
       |<---4b. retain IP1--------->|<-------->|           |
       |<..........................o==========o.........>|
       |                             |          |           |
       |<---5. MPTCP (s/IP1/IP2)------------------------->|
       |<..........................o....................>|
       |                             |          |           |
       |<---6. release IP1--------->|<-------->|           |
       |                             |   ===X== |           |
       |                             |          |           |
```

Figure 1. Coordinated use of MPTCP and Proxy Mobile IP.

Step 1:

Mobile host attaches to source gateway (s-GW) and configures an IP address (IP1).

Step 2:

Mobile host sets up an end-to-end TCP flow with a corresponding host using IP1 as its local IP address.

Step 3:

Mobile host attaches to target gateway’s (t-GW) radio network.

Step 4a:

Mobile host obtains a new IP address from t-GW (IP2) and configures that address on its IP stack.

Step 4b:

In parallel with the previous step, mobile host requests the network to continue using its previously allocated IP address (IP1). This

request results in signaling between the t-GW and s-GW, and setting up a forwarding tunnel between the two routers. The end-to-end flow continues using IP1 on the mobile host’s end. The IP packets are forwarded between the end-points via the s-GW and t-GW.

Step 5:

Mobile host updates its corresponding host to switch the TCP flow from IP1 to IP2 using MPTCP, given that both IP addresses are available to the mobile host and the latter one is preferable for optimal network use. The TCP flow gets updated with the new local IP address for the mobile host, and previously allocated IP address (IP1) and inter-GW tunnel become redundant.

Step 6:

Mobile host requests the network to release the previously-allocated IP address (IP1). Inter-GW signaling removes the associated tunnel and forwarding state.

This example illustrates how the mobile host utilizes MPTCP as its primary mobility protocol for its optimized data-path management benefit and engages Proxy Mobile IP transiently as a secondary solution for achieving seamless handovers.

4.2. IP Mobility Orchestrator

The functional entity in charge of the coordinated execution of multiple mobility protocols is called IP Mobility Orchestrator. The Mobility Orchestrator resides on the mobile host and performs the following roles:

- Discovering host mobility capabilities: Finding out the mobility protocols implemented on the host stack, including the capabilities of individual applications.

- Discovering network mobility capabilities: Finding out whether the IP/sub-IP solutions supported by the network.

- Discovering corresponding host mobility capabilities: Finding out the mobility protocols implemented on the corresponding host stack.

- Selecting primary and secondary mobility protocols: Deciding which protocols to engage for a given flow between the mobile host and its corresponding host based on the capabilities of mobile host, access network, and corresponding host.

- Coordinated execution of primary and secondary mobility protocols: Controlling the execution of the primary and secondary mobility protocols in response to IP handovers.

4.3. Call Flow

A more detailed call flow is depicted in Figure 2.

```
  Mobile                                                Corresponding
   Host                          t-GW    s-GW    DNS   Host
    |                              |       |      |     |
    * 1. discover host mob.cap.    |       |      |     |
    |                              |       |      |     |
    * 2. attach to s-GW            |       |      |     |
    |                              |       |      |     |
    |<--3a. configure IP1 --------------->|      |     |
    |                              |       |      |     |
    |<--3b. discover access.net.cap------>|      |     |
    |                              |       |      |     |
    * 4. app attempts connection   |       |      |     |
    |                              |       |      |     |
    |<--5a. resolve IP@ of cor.host------------->|     |
    |                              |       |      |     |
    |<--5b. discover cor.host mob.cap----------->|     |
    |                              |       |      |     |
    * 6. select mob. protocols     |       |      |     |
    |                              |       |      |     |
    |<..7. start e2e IP flow .............o...........>|
    |                              |       |      |     |
    * 8. attach to t-GW            |       |      |     |
    |                              |       |      |     |
    |<--9a. discover acc.net.cap->|       |      |     |
    |                              |       |      |     |
    |<--9b. configure IP2-------->|       |      |     |
    |                              |       |      |     |
    |<--9c. retain IP1----------->|<----->|      |     |
    |<...........................o=======o...........>|
    |                              |       |      |     |
    |<--10. MPTCP (s/IP1/IP2)------------------------->|
    |<...........................o...................>|
    |                              |       |      |     |
    |<--11. release IP1---------->|<----->|      |     |
    |                              | ==X== |      |     |
    |                              |       |      |     |
```

Figure 2. Use of MPTCP and Proxy Mobile IP (detailed).

Step 1:

Orchestrator discovers the mobility protocols implemented on the host stack ({MPTCP} in this example).

Step 2:

Mobile host attaches to source gateway’s (s-GW) radio network.

Step 3a:

Mobile host configures an IP address (IP1).

Step 3b:

Orchestrator discovers the mobility protocols supported by the access network ({Proxy Mobile IP-based access network anchoring} in this example).

Step 4:

An application running on the mobile host attempts to establish communication with a corresponding host.

Step 5a:

Mobile host resolves the IP address of the corresponding host in response to the associated API call (e.g., getaddrinfo()) from the application.

Step 5b:

Orchestrator discovers the mobility protocols supported by the corresponding host by using DNS ({MPTCP} in this example).

Step 6:

Orchestrator selects the primary and secondary mobility protocols for the flow between the mobile host and the corresponding host based on the discovered mobility capabilities of the mobile host, the access network, and the corresponding host (MPTCP and Proxy Mobile IP-based access network anchoring, respectively).

Step 7:

Given that MPTCP is the primary mobility protocol, the Orchestrator allows the application to bind to IP1 (a local/unanchored/nomadic IP address) and start the data flow.

Step 8:

Mobile host attaches to target gateway’s (t-GW) radio network.

Step 9a:

Orchestrator discovers the mobility protocols supported by the access network ({Proxy Mobile IP-based access network anchoring} in this example).

Step 9b:

Orchestrator requests configuration of a local IP address (IP2), given that it can be utilized by the primary mobility protocol, MPTCP.

Step 9c:

Orchestrator issues a request to the access network for retaining IP1, given that both the source and target (now serving) networks can support access network anchoring. This results in forwarding tunnel setup between the s-GW and the t-GW, and the flow continuing to use IP1 through a data-path that traverses both s-GW and t-GW.

Step 10:

Orchestrator triggers the MPTCP to update its corresponding host to switch the TCP flow from IP1 to IP2 using MPTCP, given that both IP addresses are available to the mobile host and the latter one is preferable for optimal network use. The TCP flow gets updated with the new local IP address for the mobile host, and previously allocated (anchored) IP address (IP1) and inter-GW tunnel become redundant.

Step 11:

Orchestrator requests the network to release the anchored IP address (IP1). Inter-GW signaling removes the associated tunnel and forwarding state.

4.4. Mobility Protocol Selection Algorithm

The following pseudocode describes how the Orchestrator selects primary and secondary mobility protocols when an application attempts to initiate a new flow. This algorithm is run on a per-flow basis.

```
      If there is an above-IP protocol common to both the mobile and
         corresponding host for the given flow type

         Select one of the common protocols as Primary Mobility Protocol

         If access network supports IP or sub-IP protocols

             Select one as Secondary Mobility Protocol

         Else

             There is no Secondary Mobility Protocol

     Else

         If network supports IP or sub-IP protocols

             Select one as Primary Mobility Protocol

             There is no Secondary Mobility Protocol

         Else

             There is no Primary&Secondary Mobility Protocol
```

4.5. Handover Algorithm

The following pseudocode describes how the Orchestrator coordinates the execution of the primary and secondary mobility protocols at the time of IP handovers. This algorithm is run at system-level on the mobile host.

```
If any mobility protocol is used

    If only a IP/sub-IP protocol is used

        Request IP address anchoring

    Else

        If only above-IP primary protocols used w/o any secondary
           protocols

            Release the old IP address from old GW

            Configure a new IP address from serving GW

            For each primary mobility protocol

                Execute primary protocol handover using new IP addr.

        Else /* mix of IP/sub-IP and above-IP protocols used */

           Request IP address anchoring with old GW

           Configure a new IP address from serving GW

           For each primary mobility protocol

                Execute primary protocol handover using new IP addr.

           If no flow using IP/sub-IP as primary mobility protocol

                Release the old IP address from old GW

Else /* no mobility protocol is used */

    Release the old IP address from old GW

    Configure a new IP address from serving GW
```

5. Security Considerations

TBD

6. IANA Considerations

TBD

7. References

7.1. Normative References

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

7.2. Informative References

[GTP] "3GPP Evolved Packet System (EPS); Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C); Stage 3", TS 29.274 , June 2014.

[I-D.ietf-dmm-requirements] Chan, A., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", draft-ietf-dmm-requirements-17 (work in progress), June 2014.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, June 2006.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

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

[RFC5563] Leung, K., Dommety, G., Yegani, P., and K. Chowdhury, "WiMAX Forum / 3GPP2 Proxy Mobile IPv4", RFC 5563, February 2010.

[RFC5944] Perkins, C., "IP Mobility Support for IPv4, Revised", RFC 5944, November 2010.

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

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, January 2013.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

Authors' Addresses

Alper Yegin
Samsung
Istanbul
Turkey

Email: alper.yegin@partner.samsung.com

Jungshin Park
Samsung
Suwon
South Korea

Email: shin02.park@samsung.com

Kisuk Kweon
Samsung
Suwon
South Korea

Email: kisuk.kweon@samsung.com

Jinsung Lee
Samsung
Suwon
South Korea

Email: js81.lee@samsung.com