DMM Working Group   CJ. Bernardos
Internet-Draft   UC3M
Intended status: Standards Track   JC. Zuniga
Expires: November 30, 2017   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:

- 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".

- 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.

- 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.

- 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:

- o 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.

- o 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 WG                                                     S. Gundavelli
Internet-Draft                                                     Cisco
Intended status: Informational                                  S. Jeon
Expires: March 2, 2018                          Sungkyunkwan University
                                                        August 29, 2017
```

DMM Deployment Models and Architectural Considerations
draft-ietf-dmm-deployment-models-02.txt

Abstract

This document identifies the deployment models for Distributed Mobility Management architecture.

Status of this Memo

Copyright Notice

Table of Contents

1. Overview

One of the key aspects of the Distributed Mobility Management (DMM) architecture is the separation of control plane (CP) and data plane (DP) functions of a network element. While data plane elements continue to reside on customized networking hardware, the control plane resides as a software element in the cloud. This is usually referred to as CP-DP separation and is the basis for the IETF’s DMM Architecture. This approach of centralized control plane and distributed data plane allows elastic scaling of control plane and efficient use of common data plane that is agnostic to access architectures.

This document identifies the functions in the DMM architecture and the supported deployment models.

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 are to interpreted as defined in [RFC6275], [RFC5213], [RFC5844], [RFC7333], [RFC7429], [I-D.ietf-sfc-nsh] and [I-D.ietf-dmm-fpc-cpdp]. Additionally, this document uses the following terms:

Home Control-Plane Anchor (H-CPA)

The Home-CPA function hosts the mobile node’s mobility session. There can be more than one mobility session for a mobile node [MN] and those sessions may be anchored on the same or different Home-CPA’s. The home-CPA will interface with the home-dpa for managing the forwarding state.

Home Data Plane Anchor (Home-DPA)

The Home-DPA is the topological anchor for the mobile node’s IP address/prefix(es). The Home-DPA is chosen by the Home-CPA on a session-basis. The Home-DPA is in the forwarding path for all the mobile node’s IP traffic.

Access Control Plane Node (Access-CPN)

The Access-CPN is responsible for interfacing with the mobile node’s Home-CPA and with the Access-DPN. The Access-CPN has a protocol interface to the Home-CPA.

Access Data Plane Node (Access-DPN)

The Access-DPN function is hosted on the first-hop router where the mobile node is attached. This function is not hosted on a layer-2 bridging device such as a eNode(B) or Access Point.

3. DMM Architectural Overview

Following are the key goals of the Distributed Mobility Management architecture.

1. Separation of control and data Plane

2. Aggregation of control plane for elastic scaling

3. Distribution of the data plane for efficient network usage

4. Elimination of mobility state from the data plane

5. Dynamic selection of control and data plane nodes

6. Enabling the mobile node with network properties

7. Relocation of anchor functions for efficient network usage

3.1. DMM Service Primitives

The functions in the DMM architecture support a set of service primitives. Each of these service primitives identifies a specific service capability with the exact service definition. The functions in the DMM architecture are required to support a specific set of service primitives that are mandatory for that service function. Not all service primitives are applicable to all DMM functions. The below table identifies the service primitives that each of the DMM function SHOULD support. The marking "X" indicates the service primitive on that row needs to be supported by the identified DMM function on the corresponding column; for example, the IP address management must be supported by Home-CPA function.

Service Primitive	H-CPA	H-DPA	A-CPN	A-DPN	MC	RC
IP Management	X				X	
IP Anchoring		X				
MN Detect			X	X		
Routing		X		X		
Tunneling		X		X		
QoS Enforcement		X		X		
FPC Client	X		X		X	
FPC Agent		X		X		X
NSH Classifier		X		X		

Figure 1: Mapping of DMM functions

3.2. DMM Functions and Interfaces

3.2.1. Home Control-Plane Anchor (H-CPA):

The Home-CPA function hosts the mobile node's mobility session. There can be more than one mobility session for a mobile node and those sessions may be anchored on the same or different Home-CPA's. The home-CPA will interface with the homd-dpa for managing the forwarding state.

There can be more than one Home-CPA serving the same mobile node at a given point of time, each hosting a different control plane session.

The Home-CPA is responsible for life cycle management of the session, interfacing with the policy infrastructure, policy control and interfacing with the Home-DPA functions.

The Home-CPA function typically stays on the same node. In some special use-cases (Ex: Geo-Redundancy), the session may be migrated to a different node and with the new node assuming the Home-CPA role for that session.

3.2.2. Home Data-Plane Anchor (H-DPA):

The Home-DPA is the topological anchor for the mobile node's IP address/prefix(es). The Home-DPA is chosen by the Home-CPA/MC on a session-basis. The Home-DPA is in the forwarding path for all the mobile node's IP traffic.

As the mobile node roams in the mobile network, the mobile node's access-DPN may change, however, the Home-DPA does not change, unless the session is migrated to a new node.

The Home-DPA interfaces with the Home-CPA/MC for all IP forwarding and QoS rules enforcement.

The Home-DPA and the Access-DPN functions may be collocated on the same node.

3.2.3. Access Control Plane Node (Access-CPN)

The Access-CPN is responsible for interfacing with the mobile node's Home-CPA and with the Access-DPN. The Access-CPN has a protocol interface to the Home-CPA.

The Access-CPN is responsible for the mobile node's Home-CPA selection based on: Mobile Node's Attach Preferences, Access and Subscription Policy, Topological Proximity and Other Considerations.

The Access-CPN function is responsible for MN's service authorization. It will interface with the access network authorization functions.

3.2.4. Access Data Plane Node (Access-DPN)

The Access-DPN function is hosted on the first-hop router where the mobile node is attached. This function is not hosted on a layer-2 bridging device such as a eNode(B) or Access Point.

The Access-DPA will have a protocol interface to the Access-CPA.

The Access-DPN and the Home-DPA functions may be collocated on the same node.

3.2.5. DMM Function Mapping to other Architectures

Following table identifies the potential mapping of DMM functions to protocol functions in other system architectures.

FUNCTION	PMIPv6	MIPv6	IPsec	3GPP	Broadband
Home-CPA	LMA-CPA	HA-CPA	IKE-CPA	PGW-CPA	BNG-CPA
Home-DPA	LMA-DPA	HA-DPA	IKE-DPA	PGW-DPA	BNG-DPA
Access-CPN	MAG-CPN	-	-	SGW-CPN	RG-CPN
Access-DPN	MAG-DPN	-	-	SGW-DPN	RG-DPN

Figure 2: Mapping of DMM functions

4. Deployment Models

This section identifies the key deployment models for the DMM architecture.

4.1. Model-1: Split Home Anchor Mode

In this model, the control and the data plane functions of the home anchor are separated and deployed on different nodes. The control plane function of the Home anchor is handled by the Home-CPA and where as the data plane function is handled by the Home-DPA. In this model, the access node operates in the legacy mode with the integrated control and user plane functions.

The FPC interface defined in [I-D.ietf-dmm-fpc-cpdp] allows the control plane functions to interact with the data plane for the subscriber’s forwarding state management.

```
                          +============+
                          |   Policy   |
             . . . . . . .|  Function  |. . . . . . .
             .            +============+            .
             .                                      .
             .                                      .
       +============+       {PMIPv6/GTP}      +============+
       |            |- - - - - - - - - - - - -|  Home-CPA  |
       |            |                         +============+
       |            |                                 .
       |            |                                 .  FPC
       | Access Node|                                 .
       |            |                                 .
       | (CPN + DPN)|                                 .
       |            |                         +============+
       |   Legacy   |. . . . . . . . . . . . .|  Home-DPA  |
       +============+   UP {Tunnel/Route}     +============+
              .
              .
             +--+
             |MN|
             +--+
```

                  Figure 3: Split Home Anchor Mode

4.2. Model-2: Seperated Control and User Plane Mode

In this model, the control and the data plane functions on both the home anchor and the access node are seperated and deployed on different nodes. The control plane function of the Home anchor is handled by the Home-CPA and where as the data plane function is handled by the Home-DPA. The control plane function of the access node is handled by the Access-CPN and where as the data plane function is handled by the Access-DPN.

The FPC interface defined in [I-D.ietf-dmm-fpc-cpdp] allows the control plane functions of the home and access nodes to interact with the respective data plane functions for the subscriber’s forwarding state management.

```
                          +============+
                          |   Policy   |
             . . . . . . .|  Function  |. . . . . . .
             .            +============+            .
             .                                      .
             .                                      .
             .                                      .
             .                                      .
       +============+    {PMIPv6/GTP}       +============+
       | Access-CPN |- - - - - - - - - - - -|  Home-CPA  |
       +============+                       +============+
             .                                      .
             .  FPC                                 .  FPC
             .                                      .
             .                                      .
             .                                      .
       +============+                       +============+
       | Access-DPN |. . . . . . . . . . .  |  Home-DPA  |
       +============+   UP {Tunnel/Route}   +============+
              .
              .
             [MN]
```

Figure 4: Seperated Control and User Plane Mode

4.3. Model-3: Centralized Control Plane Mode

In this model, the control-plane functions of the home and the access nodes are collapsed. This is a flat architecture with no signaling protocol between the access node and home anchors. The interface between the Home-CPA and the Access-DPN is internal to the system.

The FPC interface defined in [I-D.ietf-dmm-fpc-cpdp] allows the mobility controller to interact with the respective data plane functions for the subscriber’s forwarding state management.

```
                    +=======================+     +============+
                    | Home-CPA + Access-CPN |     |  Policy    |
                    |                       |-----| Function   |
                    +=======================+     +============+
                               .
                            .     .
                         .           .
                   FPC .               .  FPC
                     .                    .
                   .                        .
          +============+                    +============+
          | Access-DPN |. . . . . . .  . . . .|  Home-DPA  |
          +============+ UP {Tunnel/Route}  +============+
                .
                .
               [MN]
```

Figure 5: Centralized Control Plane Mode

4.4. Model-4: Data Plane Abstraction Mode

In this model, the data plane network is completely abstracted from the control plane. There is a new network element, Routing Controller which abstracts the entire data plane network and offers data plane services to the control plane functions. The control plane functions, Home-CPA and the Access-CPN interface with the Routing Controller for the forwarding state management.

The FPC interface defined in [I-D.ietf-dmm-fpc-cpdp] allows the Home-CPA and Access-CPN functions to interface with the Routing Controller for subscriber’s forwarding state management.

```
                          +============+
                          |   Policy   |
             . . . . . . .|  Function  |. . . . . . .
             .            +============+            .
             .                                      .
             .                                      .
             .                                      .
       +============+    {PMIPv6/GTP}        +============+
       | Access-CPN |- - - - - - - - - - - - |  Home-CPA  |
       +============+                        +============+
             .                                      .
             .                                      .
             .                                      .
             .           +============+             .
             . . . . . . |  Routing   | . . . . . . .
                         | Controller |
                         +============+
                                .
                             .     .
                          .          .  BGP/Others
                        .              .
                      .                   .
                    .                       .
          +============+                    +============+
          | Access-DPN |. . . . . .  . . . .|  Home-DPA  |
          +============+ UP {Tunnel/Route}  +============+
                .
                .
               [MN]
```

Figure 6: Data Plane Abstraction Mode

4.5. On-Demand Control Plane Orchestration Mode

In this model, there is a new function Mobility Controller which manages the orchestration of Access-CPN and Home-CPA functions. The Mobility Controller allocates the Home-CPA and Access-DPN

```
   + - - - - - - - - - - - - - - - - - - - - - - - - - - -+
   |    +----------+     +----------+     +----------+    |
        |Access-CPN|     |Access-CPN|     |Access-CPN|
   |    +----------+     +----------+     +----------+    |

   |    +----------+     +----------+     +----------+    |
        | Home-CPA |     | Home-CPA |     | Home-CPA |
   |    +----------+     +----------+     +----------+    |
   + - - - - - - - - - - - - - - - - - - - - - - - - - - -+
              .                 .
              .                 .
              .                 .
              .          +============+     +============+
              .          |   Mobility |     |  Policy    |
              .          | Controller |-----| Function   |
              .          +============+     +============+
              .
              .
              .
              .          +============+
              . . . . . .|  Routing   |
                         | Controller |
                         +============+
                                .
                                .
                                .
   + - - - - - - - - - - - - - - - - - - - - - - - - - - -+
   |    +----------+     +----------+     +----------+    |
        |Access-DPN|     |Access-DPN|     |Access-DPN|
   |    +----------+     +----------+     +----------+    |

   |    +----------+     +----------+     +----------+    |
        | Home-DPA |     | Home-DPA |     | Home-DPA |
   |    +----------+     +----------+     +----------+    |
   + - - - - - - - - - - - - - - - - - - - - - - - - - - -+
```

Figure 7: On-Demand CP Orchestration Mode

5. IANA Considerations

This document does not require any IANA actions.

6. Security Considerations

The control-plane messages exchanged between a Home-CPA and the Home-DPA must be protected using end-to-end security associations with data-integrity and data-origination capabilities.

IPsec ESP in transport mode with mandatory integrity protection should be used for protecting the signaling messages. IKEv2 should be used to set up security associations between the Home-CPA and Home-DPA.

There are no additional security considerations other than what is presented in the document.

7. Work Team

This document reflects contributions from the following work team members:

Younghan Kim

younghak@ssu.ac.kr

Vic Liu

liuzhiheng@chinamobile.com

Danny S Moses

danny.moses@intel.com

Marco Liebsch

liebsch@neclab.eu

Carlos Jesus Bernardos Cano

cjbc@it.uc3m.es

8. Acknowledgements

This document is a result of DMM WT#4 team discussions and ideas taken from several DMM WG presentations and documents including, draft-sijeon-dmm-deployment-models, draft-liu-dmm-deployment-scenario and others. The work teams would like to thank the authors of these documents and additionally the discussions in DMM Working group that

helped shape 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, DOI 10.17487/ RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

9.2. Informative References

[I-D.ietf-dmm-fpc-cpdp] Matsushima, S., Bertz, L., Liebsch, M., Gundavelli, S., Moses, D., and C. Perkins, "Protocol for Forwarding Policy Configuration (FPC) in DMM", draft-ietf-dmm-fpc-cpdp-07 (work in progress), March 2017.

[I-D.ietf-sfc-nsh] Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", draft-ietf-sfc-nsh-19 (work in progress), August 2017.

[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>.

[RFC5844] Wakikawa, R. and S. Gundavelli, "IPv4 Support for Proxy Mobile IPv6", RFC 5844, DOI 10.17487/RFC5844, May 2010, <https://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, <https://www.rfc-editor.org/info/rfc6275>.

[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>.

Authors' Addresses

Sri Gundavelli
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sgundave@cisco.com

Seil Jeon
Sungkyunkwan University
2066 Seobu-ro, Jangan-gu
Suwon, Gyeonggi-do
Korea

Email: seiljeon@skku.edu

DMM
Internet-Draft
Intended status: Informational
Expires: January 4, 2018

H. Chan, Ed.
X. Wei
Huawei Technologies
J. Lee
Sangmyung University
S. Jeon
Sungkyunkwan University
A. Petrescu
CEA, LIST
F. Templin
Boeing Research and Technology

July 3, 2017

Distributed Mobility Anchoring
draft-ietf-dmm-distributed-mobility-anchoring-06

Abstract

This document defines distributed mobility anchoring in terms of the different configurations, operations and parameters of mobility functions to provide different IP mobility support for the diverse mobility needs in 5G Wireless and beyond. A network may be configured with distributed mobility anchoring functions according to the needs of mobility support. In the distributed mobility anchoring environment, multiple anchors are available for mid-session switching of an IP prefix anchor. To start a new flow or to handle a flow not requiring IP session continuity as a mobile node moves to a new network, the flow can be started or re-started using a new IP address configured from the new IP prefix which is anchored to the new network. For a flow requiring IP session continuity, the anchoring of the prior IP prefix may be moved to the new network. The mobility functions and their operations and parameters are general for different configurations. The mobility signaling may be between anchors and nodes in the network in a network-based mobility solution. It may also be between the anchors and the mobile node in a host-based solution. The mobile node may be a host, but may also be a router carrying a network requiring network mobility support.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

A key requirement in distributed mobility management [RFC7333] is to enable traffic to avoid traversing a single mobility anchor far from an optimal route. This draft defines different configurations, functional operations and parameters for distributed mobility anchoring and explains how to use them to make the route changes to avoid unnecessarily long routes.

Companion distributed mobility management documents are already addressing the architecture and deployment [I-D.ietf-dmm-deployment-models], source address selection [I-D.ietf-dmm-ondemand-mobility], and control-plane data-plane signaling [I-D.ietf-dmm-fpc-cpdp]. A number of distributed mobility solutions have also been proposed, for example, in [I-D.seite-dmm-dma], [I-D.bernardos-dmm-cmip], [I-D.bernardos-dmm-pmip], [I-D.sarikaya-dmm-for-wifi], [I-D.yhkim-dmm-enhanced-anchoring], and [I-D.matsushima-stateless-uplane-vepc]. Yet in 5G Wireless and beyond, the mobility requirements are diverse, and IP mobility support is no longer by default with a one-size-fit-all solution. In different networks, different kinds of mobility support are possible depending on the needs. In designing mobility solutions, it may not always be obvious on how to best configure and use only the needed mobility functions to provide the specific mobility support. This document aims at filling such background.

Distributed mobility anchoring employs multiple anchors in the data plane. In general, control plane functions may be separate from data plane functions and be centralized but may also be co-located with the data plane functions at the distributed anchors. Different configurations of distributed mobility anchoring are described in Section 3.1. For instance, the configurations for network-based mobility support in a flat network, for network-based mobility support in a hierarchical network, for host-based mobility support, and for network mobility basic support are described respectively in Section 3.1.1, Section 3.1.2, Section 3.1.3 and Section 3.1.4. Required operations and parameters for distributed mobility anchoring are presented in Section 3.2. For instance, location management is described in Section 3.2.1, forwarding management is described in Section 3.2.2.

As an MN attaches to an access router and establishes a link between them, a /64 IPv6 prefix anchored to the router may be assigned to the link for exclusive use by the MN [RFC6459]. The MN may then configure a global IPv6 address from this prefix and use it as the source IP address in a flow to communicate with with its correspondent node (CN). When there are multiple mobility anchors, an address selection for a given flow is first required before the flow is initiated. Using an anchor in an MN's network of attachment has the advantage that the packets can simply be forwarded according to the forwarding table. However, after the flow has been initiated, the MN may later move to another network, so that the IP address no longer belongs to the current network of attachment of the MN.

Whether the flow needs IP session continuity will determine how to ensure that the IP address of the flow will be anchored to the new network of attachment. If the ongoing IP flow can cope with an IP prefix/address change, the flow can be reinitiated with a new IP address anchored in the new network as shown in Section 4.1. On the other hand, if the ongoing IP flow cannot cope with such change, mobility support is needed as shown in Section 4.2. A network supporting a mix of flows both requiring and not requiring IP mobility support will need to distinguish these flows. The guidelines for the network to make such a distinction are described in Section 4.1.1. The general guidelines for such network to provide IP mobility support are described in Section 4.2.1.

Specifically, IP mobility support can be provided by relocating the anchoring of the IP prefix/address of the flow from the home network of the flow to the new network of attachment. The basic case may be with network-based mobility for a flat network configuration described in Section 5.1 with the guidelines described in Section 5.1.1. This case is discussed further with a centralized control plane in Section 5.2 with additional guidelines described in

Section 5.2.1. A level of hierarchy of nodes may then be added to the network configuration as described in Section 5.3 with additional guidelines described in Section 5.3.1. Local Mobility in such hierarchical network is described in Section 5.4 with additional guidelines described in Section 5.4.1. Network mobiltiy example is described in Section 5.5 with additional guidelines described in Section 5.5.1.

2. Conventions and Terminology

The key words "MUST", "MUST NOT", "GLUIRED", "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 (MIPv6) base specification [RFC6275], the Proxy Mobile IPv6 (PMIPv6) specification [RFC5213], the "Mobility Related Terminologies" [RFC3753], and the DMM current practices and gap analysis [RFC7429]. These include 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 terms:

Home network of an application session or a home address: the network that has assigned the HoA used as the session identifier by the application running in an MN. The MN may be running multiple application sessions, and each of these sessions can have a different home network.

Anchoring (an IP prefix/address): An IP prefix, i.e., Home Network Prefix (HNP), or address, i.e., HoA, assigned for use by an MN is topologically anchored to an anchor node when the anchor node is able to advertise a connected route into the routing infrastructure for the assigned IP prefix.

Location Management (LM) function: that keeps and manages the network location information of an MN. The location information may be a binding of the advertised IP address/prefix, e.g., HoA or HNP, to the IP routing address of the MN or of a node that can forward packets destined to the MN.

When the MN is a mobile router (MR) carrying a mobile network of mobile network nodes (MNN), the location information will also include the mobile network prefix (MNP), which is the aggregate IP prefix delegated to the MR to assign IP prefixes for use by the MNNs in the mobile network.

In a client-server protocol model, location query and update messages may be exchanged between a Location Management client (LMc) and a Location Management server (LMs), where the location information can be updated to or queried from the LMs. Optionally, there may be a Location Management proxy (LMp) between LMc and LMs.

With separation of control plane and data plane, the LM function is in the control plane. It may be a logical function at the control plane node, control plane anchor, or mobility controller.

It may be distributed or centralized.

Forwarding Management (FM) function: packet interception and forwarding to/from the IP address/prefix assigned for use by the MN, based on the internetwork location information, either to the destination or to some other network element that knows how to forward the packets to their destination.

This function may be used to achieve traffic indirection. With separation of control plane and data plane, the FM function may split into a FM function in the data plane (FM-DP) and a FM function in the control plane (FM-CP).

FM-DP may be distributed with distributed mobility management. It may be a function in a data plane anchor or data plane node.

FM-CP may be distributed or centralized. It may be a function in a control plane node, control plane anchor or mobility controller.

3. Distributed Mobility Anchoring

3.1. Configurations for Different Networks

The mobility functions may be implemented in different configurations of distributed mobility anchoring in architectures separating the control and data planes. The separation described in [I-D.ietf-dmm-deployment-models] has defined the home control plane anchor (Home-CPA), home data plane anchor (Home-DPA), access control plane node (Access-CPN), and access data plane node (Access-DPN), which are respectively abbreviated as CPA, DPA, CPN, and DPN here.

Different networks may have different configurations in distributed mobility anchoring.

The configurations also differ depending on the desired mobility supports: network-based mobility support for a flat network in Section 3.1.1, network-based mobility support for a hierarchical network in Section 3.1.2, host-based mobility support in Section 3.1.3, and NEtwork MObility (NEMO) based support in Section 3.1.4.

3.1.1. Network-based Mobility Support for a Flat Network

Figure 1 show the configurations of network-based distributed mobility management for a flat network.

The features in Figure 1 are:

dmm:1 There are multiple instances of DPA, each with an FM-DP function.

dmm:2 The control plane may either be distributed (not shown) or centralized. The CPA and DPA may co-locate or may be separate. When the CPA, each with an FM-CP function, is co-located with the distributed DPA there will be multiple instances of the co-located CPA and DPA (not shown).

dmm:3 An IP prefix/address IP1, which is anchored to the DPA with the IP prefix/address IPa1, is assigned for use by an MN. The MN uses IP1 to communicate with a CN not shown in the figure. The flow of this communication session is shown as flow(IP1, ...), meaning it uses IP1 and other parameters.

```
              _____________  Network
          ___/             \___________
         /      +-----+                \___
        (       |LMs  |    Control         \
       /        +-.---+    plane            \
      /  +--------.---+    functions         \
     (   |CPA:    .   |    in the             )
     (   |FM-CP, LMc  |    network            )
     (   +------------+                        \
    /          . .                              \
   (           .     .                           )
   (           .         .                       )
   (           .             .                   \
    \    +------------+ +------------+Distributed )
     (   |DPA(IPa1):  | |DPA(IPa2):  |DPA's       )
     (   |anchors IP1 | |anchors IP2 |          _/
      \  |FM-DP       | |FM-DP       | etc.    /
       \ +------------+ +------------+        /
        \___                Data plane  _____/
            \______         functions  /
                   \__________________/

         +------------+
         |MN(IP1)     | Mobile node attached
         |flow(IP1,..)| to the network
         +------------+
```

Figure 1. Configurations of network-based mobility management for a flat network to which MN is attached. The mobility management functions in the network are LMs in the control plane, LMc at CPA, and FM-DP at DPA.

In Figure 1, the LM function is split into a separate server LMs and a client LMc at the CPA. Then, the LMs may be centralized whereas the LMc may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

In a special case (not shown), LMs and LMc may co-locate.

3.1.2. Network-based Mobility Support for a Hierarchical Network

Figure 2 shows the configurations of network-based mobility management for a hierarchical network.

```
       +-----+
       |LMs  |
       +-.---+
+--------.---+
|CPA:    .   |
|FM-CP, LMp  |
+------------+
      .   .
      .         .
      .                   .
      .                             .
+------------+                        +------------+ Distributed
|DPA(IPa1):  |                        |DPA(IPa2):  | DPA’s
|anchors IP1 |                        |anchors IP2 |                etc.
|FM-DP       |                        |FM-DP       |
+------------+                        +------------+

+------------+
|CPN:        |
|FM-CP, LMc  |
+------------+
      . .
      .     .
      .         .
      .             .Distributed DPN’s
+------------+ +------------+         +------------+ +------------+
|DPN(IPn11): | |DPN(IPn12): |         |DPN(IPn21): | |DPN(IPn22): |
|FM-DP       | |FM-DP       | etc.    |FM-DP       | |FM-DP       | etc.
+------------+ +------------+         +------------+ +------------+

+------------+Mobile node             +------------+Mobile node
|MN(IP1)     |using IP1               |MN(IP2)     |using IP1
|flow(IP1,..)|anchored to             |flow(IP2,..)|anchored to
+------------+DPA(IPa1)               +------------+DPA(IPa2)
```

Figure 2. Configurations of network-based mobility management for a hierarchical network to which MN is attached. The mobility management functions in the network include a separate LMs, FM-CP and LMp at CPA, FM-DP at DPA; FM-CP and LMc at CPN, FM-DP at DPN.

In addition to the dmm feature already described in Figure 1, Figure 2 shows that there may be multiple instances of DPN, each with an FM-DP function, for each DPA in the hierarchy. Also when the CPN, each with an FM-CP function, is co-located with the distributed DPN there will be multiple instances of the co-located CPN and DPN (not shown).

In Figure 2 the LMs is separated out, and a proxy LMp at the CPA is added between the separate LMs and LMc at the CPN. Then, LMs may be centralized whereas the LMp may be distributed or centralized according to whether the CPA is distributed or centralized.

In a particular case (not shown), LMs and LMp may co-locate.

3.1.3. Host-based Mobility Support

Host-based mobility function configurations as variants from Figures 2 is shown in Figure 3 where the role to perform mobility functions by CPN and DPN are now taken by the MN. The MN then needs to possess the mobility functions FM and LMc.

```
       +-----+
       |LMs  |
       +-.---+
+--------.---+
|CPA:    .   |
|FM-CP, LMp  |
+------------+
      . .
      .     .
      .         .
      .             .
+------------+ +------------+ Distributed
|DPA(IPa1):  | |DPA(IPa2):  | DPA’s
|anchors IP1 | |anchors IP2 |
|FM-DP       | |FM-DP       |  etc.
+------------+ +------------+

+------------+
|MN(IP1)     |Mobile node
|flow(IP1,..)|using IP1
|FM,    LMc  |anchored to
+------------+DPA(IPa1)
```

Figure 3. Configuration of host-based mobility management. The mobility management functions in the network include LMs in control plane, FM-CP and LMp at CPA, FM-DP at DPA. The mobility management functions FM and LMc are also at the host (MN).

Figure 3 shows configurations of host-based mobility management with multiple instances of DPA for a distributed mobility anchoring environment. Figures 3 can be obtained by simply collapsing CPN, DPN and MN from the Figures 2 into the MN in Figure 3 which now possesses the mobility functions FM and LMc that were performed previously by the CPN and the DPN.

3.1.4. NEtwork MObility (NEMO) Basic Support

Figure 4 shows the configurations of NEMO basic support for a mobile router.

```
       +-----+
       |LMs  |
       +-.---+
+--------.---+
|CPA:    .   |
|FM-CP, LMp  |
+------------+
      .  .
      .       .
      .            .
      .                 .
+--------------+   +--------------+ Distributed
|DPA(IPa1):    |   |DPA(IPa2):    | DPA’s
|anchors IP1   |   |anchors IP2   |
|DHCPv6-PD IPn1|   |DHCPv6-PD IPn2|  etc.
|FM-DP         |   |FM-DP         |
+--------------+   +--------------+

+--------------+Mobile router
|MR(IP1)       |using IP1
|delegated IPn1|anchored to
|FM,    LMc    |DPA(IPa1)
+--------------+

+------------+Mobile network node
|MNN(IPn1)   |using IPn1
|flow(IPn1,.)|attached to MR(IP1)
+------------+
```

Figure 4. Configurations of NEMO basic support for a MR which is attached to a network. The mobility management functions in the network are a separate LMs, FM-CP and LMp at CPA, FM-DP at DPA. The mobility management functions FM and LMc are also at the MR to which MNN is attached.

Figure 4 shows configurations of host-based mobility management for a MR with multiple instances of DPA for a distributed mobility anchoring environment. Figure 4 can be obtained by simply changing the MN from the Figures 3 into the MR carrying a mobile network consisting of mobile network nodes (MNNs) in Figure 4.

An IP prefix/address IPn1 delegated to the MR is assigned for use by the MNN in the mobile network. The MNN uses IPn1 to communicate with a correspondent node (CN) not shown in the figure. The flow of this communication session is shown as flow(IPn1, ...), meaning it uses IPn1 and other parameters.

To enable the MR to assign the IP prefix IPn1, the DPA delegates the prefix using DHCPv6-PD to the MR.

3.2. Operations and Parameters

The operations of distributed mobility anchoring are defined in order that they might work together to produce a distributed mobility solution. The needed information is passed as mobility message parameters, which must be protected in terms of integrity. Some parameters may require a means to support privacy of an MN or MR.

The mobility needs in 5G Wireless and beyond are diverse. Therefore operations needed to enable different distributed mobility solutions in different distributed mobility anchoring configurations are extensive as illustrated below. It is however not necessary for every distributed mobility solution to exhibit all the operations listed in this section. A given distributed mobility solution may exhibit only those operations needed.

3.2.1. Location Management

An example LM design consists of a distributed database with multiple LMs servers. The location information about the prefix/address of an MN is primarily at a given LMs. Peer LMs may exchange the location information with each other. LMc may retrieve a given record or send a given record update to LMs.

Location management configurations:

LM-cfg: As shown in Section 3.1:

> LMs may be implemented at CPA, may be co-located with LMc at CPA, or may be a separate server.
>
> LMc may be at CPA, CPN, or MN.
>
> LMp may proxy between LMs and LMc.
>
> Specifically:

Location management operations and parameters:

LM-cfg:1 LMs and LMc may co-locate or may be separate, whereas LMc is implemented in CPA in a flat network with network-based mobility as shown in Figure 1 in Section 3.1.1.

LM-cfg:2 Either LMs may be a separate server with LMp implemented at CPA, or LMs may be implemented at CPA. LMc is implemented at CPN in a hierarchical network with network-based mobility as shown in Figure 2 in Section 3.1.2, at MN for host-based mobility as shown in Figure 3 in Section 3.1.3, or at MR for network mobility as shown in Figure 4 in Section 3.1.4.

LM-db: LM may manage the location information in a client-server database system.

Example LM database functions are as follows:

LM-db:1 LMc may query LMs about location information for a prefix of MN (pull).
Parameters:

- IP prefix of MN: integrity support required and privacy support may be required.

LM-db:2 LMs may reply to LMc query about location information for a prefix of MN (pull).
Parameters:

- IP prefix of MN: integrity support required and privacy support may be required,
- IP address of FM-DP/DPA/DPN to forward the packets of the flow: integrity support required.

LM-db:3 LMs may inform LMc about location information for a prefix of MN (push).
Parameters:

- IP prefix of MN: integrity support required and privacy support may be required,
- IP address of FM-DP/DPA/DPN to forward the packets of the flow: integrity support required.

This function in the PMIPv6 protocol is the Update Notification (UPN) together with the Update Notification Acknowledgment (UPA) as defined in [RFC7077].

LM-db:4 LMc may inform LMs about update location information for a prefix of MN.
Parameters:

- IP prefix of MN: integrity support required and privacy support may be required,
- IP address of FM-DP/DPA/DPN to forward the packets of the flow: integrity support required.

This function in the MIPv6 / PMIPv6 protocol is the Binding Update (BU) / Proxy Binding Update (PBU) together with the Binding Acknowledgment (BA) / Proxy Binding Acknowledgment (PBA) as defined in [RFC6275] / [RFC5213] respectively.

LM-db:5 The MN may be a host or a router. When the MN is an MR, the prefix information may include the IP prefix delegated to the MR.
Additional parameters:

- IP prefix delegated to MR: integrity support required and privacy support may be required,
- IP prefix/address of the MR to forward the packets of the prefix delegated to the MR: integrity support required.

LM-svr: The LM may be a distributed database with multiple LMs servers.

For example:

LM-svr:1 A LMs may join a pool of LMs servers.
Parameters:

- IP address of the LMs: integrity support required,
- IP prefixes for which the LMs will host the primary location information: integrity support required.

LM-svr:2 LMs may query a peer LMs about location information for a prefix of MN.
Parameters:

- IP prefix: integrity support required and privacy support may be required.

LM-svr:3 LMs may reply to a peer LMs about location information for a prefix of MN.
Parameters:

- IP prefix of MN: integrity support required and privacy support may be required,
- IP address of FM-DP/DPA/DPN to forward the packets of the flow: integrity support required.

The list above only gives the minimal set of the required parameters. In a specific mobility protocol, additional parameters should be added as needed. Examples of these additional parameters are those passed in the mobility options of the mobility header for MIPv6 [RFC6275] and for PMIPv6 [RFC5213].

3.2.2. Forwarding Management

Forwarding management configurations:

FM-cfg: As shown in Section 3.1:

FM-CP may be implemented at CPA, CPN, MN depending on the configuration chosen.

FM-DP may also be implemented at CPA, CPN, MN depending on the configuration chosen.

Specifically:

FM-cfg:1 FM-CP and FM-DP may be implemented at CPA and DPA respectively in a flat network with network-based mobility as shown in Figure 1 in Section 3.1.1.

FM-cfg:2 FM-CP may be implemented at both CPA and CPN and FM-DP is implemented at both DPA and DPN in a hierarchical network with network-based mobility as shown in Figure 2 in Section 3.1.2.

FM-cfg:3 FM-CP and FM-DP may be implemented at CPA and DPA respectively and also both implemented at MN for host-based mobility as shown in Figure 3 in Section 3.1.3.

FM-cfg:4 FM-CP and FM-DP may be implemented at CPA and DPA respectively and also both implemented at MR for network mobility as shown in Figure 4 in Section 3.1.4.

Forwarding management operations and parameters:

FM-find:1 An anchor may discover and be discovered such as through an anchor registration system as follows:

```
   FM-find:2  FM registers and authenticates itself with a centralized
              mobility controller.
              Parameters:

              - IP address of DPA and its CPA: integrity support
                required,
              - IP prefix anchored to the DPA: integrity support
                required.

              Registration reply: acknowledge of registration and echo
              the input parameters.

   FM-find:3  FM discovers the FM of another IP prefix by querying the
              mobility controller based on the IP prefix.
              Parameters:

              - IP prefix of MN: integrity support required and privacy
                support may be required.

   FM-find:4  When making anchor discovery FM expects the answer
              parameters:

              - IP address of DPA to which IP prefix of MN is anchored:
                integrity support required,
              - IP prefix of the corresponding CPA: integrity support
                required.


   FM-flow:1  The FM may be carried out on the packets to/from an MN up
              to the granularity of a flow.

   FM-flow:2  Example matching parameters are in the 5-tuple of a flow.


   FM-path:1  FM may change the forwarding path of a flow upon a change
              of point of attachment of an MN.  Prior to the changes,
              packets coming from the CN to the MN would traverse from
              the CN to the home network anchor of the flow for the MN
              before reaching the MN.  Changes are from this original
              forwarding path or paths to a new forwarding path or paths
              from the CN to the current AR of the MN and then the MN
              itself.

   FM-path:2  As an incoming packet is forwarded from the CN to the MN,
              the far end where forwarding path change begins may in
              general be any node in the original forwarding path from
              the CN to the home network DPA.  The packet is forwarded
              to the MN for host-based mobility and to a node in the
```

network which will deliver the packets to the MN for network-based mobility. The near-end is generally a DPN with a hierarchical network but may also be another node with DPA capability in a flattened network.

FM-path:3 The mechanisms to accomplish such changes may include changes to the forwarding table and indirection such as tunneling, rewriting packet header, or NAT.

Note: An emphasis in this document in distributed mobility anchoring is to explain the use of multiple anchors to avoid unnecessarily long route which may be encountered in centralized mobility anchoring. It is therefore not the emphasis of this document on which particular mechanism to choose from.

FM-path-tbl:4 The objective of forwarding table updates is to change the forwarding path so that the packets in the flow will be forwarded from the CN to the new AR instead of the home network anchor or previous AR. Each of the affected forwarding switches will need appropriate changes to its forwarding table.

Specifically, such forwarding table updates may include: (1) addition of forwarding table entries needed to forward the packets destined to the MN to the new AR; (2) deletion of forwarding table entries to forward the packets destined to the MN to the home network anchor or to the previous AR.

FM-path-tbl:5 With a centralized control plane, forwarding table updates may be achieved through messaging between the centralized control plane and the distributed forwarding switches as described above (FM-cpdp) in this section.

FM-path-tbl:6 To reduce excessive signaling, the scope of such updates for a given flow may be confined to only those forwarding switches such that only the packets sent from the "CN" to the MN will go to the new AR. Such confinement may be made when using a centralized control plane possessing a global view of all the forwarding switches.

FM-path-tbl:7 FM reverts the changes previously made to the forwarding path of a flow when such changes are no longer needed, e.g., when all the ongoing flows using an IP prefix/address requiring IP session continuity have closed.

FM-path-ind:8 Indirection forwards the incoming packets of the flow from the DPA at the far end to a DPA/DPN at the near end of indirection. Both ends of the indirection need to know the LM information of the MN for the flow and also need to possess FM capability to perform indirection.

FM-path-ind:9 The mechanism of changing the forwarding path in MIPv6 [RFC6275] and PMIPv6 [RFC5213] is tunneling. In the control plane, the FM-CP sets up the tunnel by instructing the FM-DP at both ends of the tunnel. In the data plane, the FM-DP at the start of the tunnel performs packet encapsulation, whereas the FM-DP at the end of the tunnel decapsulates the packet.

Note that in principle the ends of the indirection path can be any pair of network elements with the FM-DP function.

FM-path-ind:10 FM reverts the changes previously made to the forwarding path of a flow when such changes are no longer needed, e.g., when all the ongoing flows using an IP prefix/address requiring IP session continuity have closed. When tunneling is used, the tunnels will be torn down when they are no longer needed.

FM-cpdp: With separation of control plane function and data plane function, FM-CP and FM-DP communicate with each other. Such communication may be realized by the appropriate messages in [I-D.ietf-dmm-fpc-cpdp].

For example:

FM-cpdp:1 CPA/FM-CP sends forwarding table updates to DPA/FM-DP. Parameters:

- New forwarding table entries to add: integrity support required,
- Expired forwarding table entries to delete: integrity support required.

FM-cpdp:2 DPA/FM-DP sends to CPA/FM-CP about its status and load. Parameters:

- State of forwarding function being active or not: integrity support required,
- Loading percentage: integrity support required.

FM-CPA: The CPA possesses FM-CP function to make the changes to the forwarding path as described in FM-path, and the changes may be implemented through forwarding table changes or through indirection as described respectively in FM-path-tbl and FM-path-ind above.

The FM-CP communicates with the FM-DP using the appropriate messages in [I-D.ietf-dmm-fpc-cpdp] as described in FM-cpdp above so that it may instruct the FM-DP to perform the changed forwarding tasks.

FM-DPA: The DPA possesses FM-DP function to forward packets according to the changed forwarding path as described in FM-path, and also FM-path-tbl or FM-path-ind depending on whether forwarding table changes or indirection is used.

The FM-DP communicates with the FM-CP using the appropriate messages in [I-D.ietf-dmm-fpc-cpdp] as described in FM-cpdp above so that it may perform the changed forwarding tasks.

The operations and their parameters for the DPA to perform distributed mobility management are described below:

FM-DPA:1 The DPAs perform the needed functions such that for the incoming packets from the CN, forwarding path change by FM is from the DPA at the far end which may be at any forwarding switch (or even CN itself) in the original forwarding path to the near end DPA/DPN.

FM-DPA:2 It is necessary that any incoming packet from the CN of the flow must traverse the DPA (or at least one of the DPAs, e.g., in the case of anycast) at the far end in order for the packet to detour to a new forwarding path. Therefore a convenient design is to locate the far end DPA at a unique location which is always in the forwarding path. This is the case in a centralized mobility design where the DPA at the far end is the home network anchor of the flow.

Distributed mobility however may place the far end DPA at other locations in order to avoid unnecessarily long route.

FM-DPA:3 With multiple nodes possessing DPA capabilities, the role of FM to begin path change for the incoming packets of a flow at the home network DPA at the far end may be passed to or added to that of another DPA.

In particular, this DPA role may be moved upstream from the home network DPA in the original forwarding path from CN to MN.

FM-DPA:4 Optimization of the new forwarding path may be achieved when the path change for the incoming packets begins at a DPA where the original path and the direct IPv6 path overlap. Then the new forwarding path will resemble the direct IPv6 path from the CN to the MN.

FM-DPA-ind:5 Another mobility support employs indirection from the far end DPA to the near end DPA. Both DPAs need to be capable to performing indirection. For incoming packets from the CN to the MN, the far end DPA needs to start the indirection towards the near end DPA, which will be the receiving end of indirection. In addition, the near end DPA needs to continue the forwarding of the packet towards the MN, such as through L2 forwarding or through another indirection towards the MN.

FM-DPA-ind:6 With indirection, locating or moving the FM function to begin indirection upstream along the forwarding path from CN to MN again may help to reduce unnecessarily long paths.

FM-DPA-ind:7 Changes made by FM to establish indirection at the DPA and DPN, which are IPv6 nodes, at the ends of the path change for a flow will be reverted when the mobility support for the flow is no longer needed, e.g., when the flows have terminated.

FM-buffer: An anchor can buffer packets of a flow in a mobility event:

FM-buffer:1 CPA/FM-CP informs DPA/FM-DP to buffer packets of a flow. Trigger:

- MN leaves DPA in a mobility event.

Parameters:

- IP prefix of the flow for which packets need to be buffered: integrity support required

FM-buffer:2 CPA/FM-CP on behalf of a new DPA/FM-DP informs the CPA/FM-CP of the prior DPA/FM-DP that it is ready to receive any buffered packets of a flow.
Parameters:

- Destination IP prefix of the flow’s packets: integrity support required,
- IP address of the new DPA: integrity support required.

FM-mr:1 When the MN is a mobile router (MR) the access router anchoring the IP prefix of the MR will also own the IP prefix or prefixes to be delegated to the MR. The MNNs in the network carried by the MR obtain IP prefixes from the MR.

4. IP Mobility Handling in Distributed Anchoring Environments - Mobility Support Only When Needed

IP mobility support may be provided only when needed instead of being provided by default. The LM and FM functions in the different configurations shown in Section 3.1 are then utilized only when needed.

A straightforward choice of mobility anchoring is for a flow to use the IP prefix of the network to which the MN is attached when the flow is initiated [I-D.seite-dmm-dma].

The IP prefix/address at the MN’s side of a flow may be anchored at the access router to which the MN is attached. For example, when an MN attaches to a network (Net1) or moves to a new network (Net2), an IP prefix from the attached network is assigned to the MN’s interface. In addition to configuring new link-local addresses, the MN configures from this prefix an IP address which is typically a dynamic IP address. It then uses this IP address when a flow is initiated. Packets to the MN in this flow are simply forwarded according to the forwarding table.

There may be multiple IP prefixes/addresses that an MN can select when initiating a flow. They may be from the same access network or different access networks. The network may advertise these prefixes with cost options [I-D.mccann-dmm-prefixcost] so that the mobile node may choose the one with the least cost. In addition, these IP prefixes/addresses may be of different types regarding whether mobility support is needed [I-D.ietf-dmm-ondemand-mobility]. A flow will need to choose the appropriate one according to whether it needs IP mobility support.

4.1. No Need of IP Mobility: Changing to New IP Prefix/Address

When IP mobility support is not needed for a flow, the LM and FM functions are not utilized so that the configurations in Section 3.1 are simplified as shown in Figure 5.

```
Net1                                                   Net2

+---------------+                                      +---------------+
|AR1            |             AR is changed            |AR2            |
+---------------+               ------->               +---------------+
|CPA:           |                                      |CPA:           |
|---------------|                                      |---------------|
|DPA(IPa1):     |                                      |DPA(IPa2):     |
|anchors IP1    |                                      |anchors IP2    |
+---------------+                                      +---------------+

+...............+                                      +---------------+
.MN(IP1)        .               MN moves               |MN(IP2)        |
.flow(IP1,...)  .               =======>               |flow(IP2,...)  |
+...............+                                      +---------------+
```

Figure 5. Changing to the new IP prefix/address. MN running a flow using IP1 in a network Net1 changes to running a flow using IP2 in Net2.

When there is no need to provide IP mobility to a flow, the flow may use a new IP address acquired from a new network as the MN moves to the new network.

Regardless of whether IP mobility is needed, if the flow has terminated before the MN moves to a new network, the flow may subsequently restart using the new IP address assigned from the new network.

When IP session continuity is needed, even if a flow is ongoing as the MN moves, it may still be desirable for the flow to change to using the new IP prefix configured in the new network. The flow may then close and then restart using a new IP address configured in the new network. Such a change in the IP address of the flow may be enabled using a higher layer mobility support which is not in the scope of this document.

In Figure 5, a flow initiated while the MN was using the IP prefix IP1 anchored to a previous access router AR1 in network Net1 has terminated before the MN moves to a new network Net2. After moving to Net2, the MN uses the new IP prefix IP2 anchored to a new access router AR2 in network Net2 to start a new flow. The packets may then be forwarded without requiring IP layer mobility support.

An example call flow is outlined in Figure 6.

```
 MN                    AR1           AR2                              CN
  |MN attaches to AR1:  |             |                                |
  |acquire MN-ID and profile          |                                |
  |--RS---------------->|             |                                |
  |                     |             |                                |
  |<----------RA(IP1)---|             |                                |
  |                     |             |                                |
Assigned prefix IP1     |             |                                |
IP1 address configuration             |                                |
  |                     |             |                                |
  |<-Flow(IP1,IPcn,...)-+--------------------------------------------->|
  |                     |             |                                |
  |MN detaches from AR1 |             |                                |
  |MN attaches to AR2   |             |                                |
  |                     |             |                                |
  |--RS------------------------------>|                                |
  |                     |             |                                |
  |<--------------RA(IP2)-------------|                                |
  |                     |             |                                |
Assigned prefix IP2     |             |                                |
IP2 address configuration             |                                |
  |                     |             |                                |
  |<-new Flow(IP2,IPcn,...)-----------+------------------------------->|
  |                     |             |                                |
```

Figure 6. Re-starting a flow to use the IP prefix assigned from the network at which the MN is attached.

4.1.1. Guidelines for IPv6 Nodes: Changing to New IP Prefix/Address

A network may not need IP mobility support. For example, a network for stationary sensors only will never encounter mobility.

The standard functions in IPv6 already include dropping the old IPv6 prefix/address and acquiring new IPv6 prefix/address when the node changes its point of attachment to a new network. Therefore, a network not providing IP mobility support at all will not need any of the functions with the mobility operations and messages described in Section 3.2.

On the other hand, a network supporting a mix of flows both requiring and not requiring IP mobility support will need the mobility functions, which it will invoke or not invoke as needed.

The guidelines for the IPv6 nodes in a network supporting a mix of flows both requiring and not requiring IP mobility support include the following:

GL-cfg:1 A network supporting a mix of flows both requiring and not requiring mobility support may take any of the configurations described in Section 3.1 and need to implement at the appropriate IPv6 nodes the mobility functions LM and FM as described respectively in LM-cfg and FM-cfg in Section 3.2 according to the configuration chosen.

GL-mix:1 These mobility functions perform some of the operations with the appropriate messages as described in Section 3.2 depending on which mobility mechanisms are being used. Yet these mobility functions must not be invoked for a flow that does not need IP mobility support so that it is necessary to be able to distinguish the needs of a flow. The guidelines for the MN and the AR are in the following.

GL-mix:2 Regardless of whether there are flows requiring IP mobility support, when the MN changes its point of attachment to a new network, it needs to configure a new global IP address for use in the new network in addition to configuring the new link-local addresses.

GL-mix:3 The MN needs to check whether a flow needs IP mobility support. This can be performed when the application is initiated. The specific method is not in the scope of this document.

GL-mix:4 The information of whether a flow needs IP mobility support is conveyed to the network such as by choosing an IP address to be provided with mobility support as described in [I-D.ietf-dmm-ondemand-mobility]. Then as the MN attaches to a new network, if the MN was using an IP address that is not supposed to be provided with mobility support, the access router will not invoke the mobility functions described in Section 3.2 for this IP address. That is, the IP address from the prior network is simply not used in the new network.

The above guidelines are only to enable distinguishing whether there is need of IP mobility support for a flow that does not. When the flow needs IP mobility support, the list of guidelines will continue in Section 4.2.1.

4.2. Need of IP Mobility

When IP mobility is needed for a flow, the LM and FM functions in Section 3.1 are utilized. The mobility support may be provided by IP prefix anchor switching to the new network to be described in Section 5 or by using other mobility management methods ([Paper-Distributed.Mobility], [Paper-Distributed.Mobility.PMIP] and [Paper-Distributed.Mobility.Review]). Then the flow may continue to use the IP prefix from the prior network of attachment. Yet some time later, the user application for the flow may be closed. If the application is started again, the new flow may not need to use the prior network’s IP address to avoid having to invoke IP mobility support. This may be the case where a dynamic IP prefix/address rather than a permanent one is used. The flow may then use the new IP prefix in the network where the flow is being initiated. Routing is again kept simpler without employing IP mobility and will remain so as long as the MN which is now in the new network has not moved again and left to another new network.

An example call flow in this case is outlined in Figure 7.

```
 MN                    AR1           AR2                              CN
  |MN attaches to AR1:  |             |                                |
  |acquire MN-ID and profile          |                                |
  |--RS---------------->|             |                                |
  |                     |             |                                |
  |<----------RA(IP1)---|             |                                |
  |                     |             |                                |
Assigned prefix IP1     |             |                                |
IP1 address configuration             |                                |
  |                     |             |                                |
  |<-Flow(IP1,IPcn,...)-+--------------------------------------------->|
  |                     |             |                                |
  |MN detach from AR1   |             |                                |
  |MN attach to AR2     |             |                                |
  |                     |             |                                |
  |--RS------------------------------>|                                |
IP mobility support such as that described in next sub-section
  |<--------------RA(IP2,IP1)---------|                                |
  |                     |             |                                |
  |<-Flow(IP1,IPcn,...)---------------+------------------------------->|
  |                     |             |                                |
Assigned prefix IP2     |             |                                |
IP2 address configuration             |                                |
  |                     |             |                                |
Flow(IP1,IPcn) terminates             |                                |
  |                     |             |                                |
  |<-new Flow(IP2,IPcn,...)-----------+------------------------------->|
  |                     |             |                                |
```

Figure 7. A flow continues to use the IP prefix from its home network after MN has moved to a new network.

4.2.1. Guidelines for IPv6 Nodes: Need of IP Mobility

The configuration guidelines of distributed mobility for the IPv6 nodes in a network supporting a mix of flows both requiring and not requiring distributed mobility support are as follows:

GL-cfg:2 Multiple instances of DPAs (at access routers) which are providing IP prefix to the MNs are needed to provide distributed mobility anchoring in an appropriate configuration such as those described in Figure 1 (Section 3.1.1) for network-based distributed mobility or in Figure 3 (Section 3.1.3) for host-based distributed mobility.

The appropriate IPv6 nodes (CPA, DPA, CPN, DPN) have to implement the mobility functions LM and FM as described respectively in LM-cfg and FM-cfg in Section 3.2 according to the configuration chosen.

The guidelines of distributed mobility for the IPv6 nodes in a network supporting a mix of flows both requiring and not requiring distributed mobility support had begun with those given as GL-mix in Section 4.1.1 and continue as follows:

GL-mix:5 The distributed anchors may need to message with each other. When such messaging is needed, the anchors may need to discover each other as described in the FM operations and mobility message parameters (FM-find) in Section 3.2.2.

GL-mix:6 The anchors may need to provide mobility support on a per-flow basis as described in the FM operations and mobility message parameters (FM-flow) in Section 3.2.2.

GL-mix:7 Then the anchors need to properly forward the packets of the flows in the appropriate FM operations and mobility message parameters depending on the specific mobility mechanism as described in Section 3.2.2.

GL-mix:8 When using a mechanism of changing forwarding table entries, the FM operations and mobility message parameters are described in FM-path, FM-path-tbl, and FM-DPA in Section 3.2.2.

The forwarding table updates will take place at AR1, AR2, the far end DPA, and other affected switches/routers such that the packet from the CN to the MN will traverse from the far end DPA towards AR2 instead of towards AR1.

Therefore new entries to the forwarding table will be added at AR2 and the far end DPA as well as other affected switches/routers between them so that these packets will traverse towards AR2. Meanwhile, changes to the forwarding table entries will also occur at AR1 and the far end DPA as well as other affected switches/routers between them so that if these packets ever reaches any of them, they will not traverse towards AR1 but will traverse towards AR2 (see Section 3.2.2).

GL-mix:9 Alternatively when using a mechanism of indirection, the FM operations and mobility message parameters are described in FM-path, FM-path-ind, FM-DPA, and FM-DPA-ind in Section 3.2.2.

GL-mix:10 If there are in-flight packets toward the old anchor while the MN is moving to the new anchor, it may be necessary to buffer these packets and then forward to the new anchor after the old anchor knows that the new anchor is ready. Such procedures are described in the FM operations and mobility message parameters (FM-buffer) in Section 3.2.2.

5. IP Mobility Handling in Distributed Mobility Anchoring Environments - Anchor Switching to the New Network

IP mobility is invoked to enable IP session continuity for an ongoing flow as the MN moves to a new network. Here the anchoring of the IP address of the flow is in the home network of the flow, which is not in the current network of attachment. A centralized mobility management mechanism may employ indirection from the anchor in the home network to the current network of attachment. Yet it may be difficult to avoid unnecessarily long route when the route between the MN and the CN via the anchor in the home network is significantly longer than the direct route between them. An alternative is to switch the IP prefix/address anchoring to the new network.

5.1. IP Prefix/Address Anchor Switching for Flat Network

The IP prefix/address anchoring may move without changing the IP prefix/address of the flow. Here the LM and FM functions in Figure 1 in Section 3.1 are implemented as shown in Figure 8.

```
Net1                                                   Net2

+---------------+                                      +---------------+
|AR1            |                                      |AR2            |
+---------------+                                      +---------------+
|CPA:           |                                      |CPA:           |
|LM:IP1 at IPa1 |                                      |LM:IP1 at IPa2 |
|   changes to  |                                      |               |
|   IP1 at IPa2 |                                      |               |
|---------------|                                      |---------------|
|DPA(IPa1):     | anchoring of IP1 is effectively moved|DPA(IPa2):     |
|anchored IP1   |               =======>               |anchors IP2,IP1|
+---------------+                                      +---------------+

+...............+                                      +---------------+
.MN(IP1)        .               MN moves               |MN(IP2,IP1)    |
.flow(IP1,...)  .               =======>               |flow(IP1,...)  |
+...............+                                      +---------------+
```

Figure 8. IP prefix/address anchor switching to the new network. MN with flow using IP1 in Net1 continues to run the flow using IP1 as it moves to Net2.

As an MN with an ongoing session moves to a new network, the flow may preserve IP session continuity by moving the anchoring of the original IP prefix/address of the flow to the new network. One way to accomplish such move is to use a centralized routing protocol to be described in Section 5.2 with a centralized control plane.

5.1.1. Guidelines for IPv6 Nodes: Switching Anchor for Flat Network

The configuration guideline for a flat network supporting a mix of flows both requiring and not requiring IP mobility support is:

GL-cfg:3 Multiple instances of DPAs (at access routers) which are providing IP prefix to the MNs are needed to provide distributed mobility anchoring according to Figure 1 in Section 3.1 for a flat network.

The appropriate IPv6 nodes (CPA, DPA) have to implement the mobility functions LM and FM as described respectively in LM-cfg:1 or LM-cfg:2 and FM-cfg:1 in Section 3.2.

The guidelines (GL-mix) in Section 4.1.1 and in Section 4.2.1 for the IPv6 nodes for a network supporting a mix of flows both requiring and not requiring IP mobility support apply here. In addition, the following are required.

GL-switch:1 The location management provides information about which IP prefix from an AR in the original network is being used by a flow in which AR in a new network. Such information needs to be deleted or updated when such flows have closed so that the IP prefix is no longer used in a different network. The LM operations are described in Section 3.2.1.

GL-switch:2 The anchor operations to properly forward the packets for a flow are described in the FM operations and mobility message parameters in FM-path, FM-path-tbl, FM-cpdp, and FM-DPA in Section 3.2.2. If there are in-flight packets toward the old anchor while the MN is moving to the new anchor, it may be necessary to buffer these packets and then forward to the new anchor after the old anchor knows that the new anchor is ready as are described in FM-buffer in Section 3.2.2. The anchors may also need to discover each other as described also in the FM operations and mobility message parameters (FM-find).

GL-switch:3 The security policy must allow to assign to the anchor node at the new network the original IP prefix/address used by the mobile node at the previous (original) network. As the assigned original IP prefix/address is to be used in the new network, the security policy must allow the anchor node in the new network to advertise the prefix of the original IP address and also allow the mobile node to send and receive data packets with the original IP address.

GL-switch:4 The security policy must allow the mobile node to configure the original IP prefix/address used at the previous (original) network when the original IP prefix/address is assigned by the anchor node in the new network. It must also allow the mobile node to use the original IP address for the previous flow in the new network.

5.2. IP Prefix/Address Anchor Switching for Flat Network with Centralized Control Plane

An example of IP prefix anchor switching is in the case where Net1 and Net2 both belong to the same operator network with separation of control and data planes ([I-D.liu-dmm-deployment-scenario] and [I-D.matsushima-stateless-uplane-vepc]), where the controller may send to the switches/routers the updated information of the forwarding tables with the IP address anchoring of the original IP prefix/address at AR1 moved to AR2 in the new network. That is, the IP address anchoring in the original network which was advertising the prefix will need to move to the new network. As the anchoring in the new network advertises the prefix of the original IP address in the new network, the forwarding tables will be updated so that packets of the flow will be forwarded according to the updated forwarding tables.

The configurations in Figure 1 in Section 3.1 for which the FM-CP and the LM are centralized and the FM-DPs are distributed apply here. Figure 9 shows its implementation where the LM is a binding between the original IP prefix/address of the flow and the IP address of the new DPA, whereas the FM uses appropriate control plane to data plane messages.

```
Net1                                                     Net2
+----------------------------------------------------------------+
|                            CPA:                                |
|                            LM:IP1 at IPa2                      |
|                            FM-CP                               |
+----------------------------------------------------------------+

+----------------+                                      +----------------+
|AR1             |                                      |AR2             |
+----------------+                                      +----------------+
|DPA(IPa1):      | anchoring of IP1 is effectively moved|DPA(IPa2):      |
|anchored IP1    |              =======>                |anchors IP2,IP1|
+----------------+                                      +----------------+

+...............+                                      +----------------+
.MN(IP1)        .              MN moves                |MN(IP2,IP1)     |
.flow(IP1,...)  .              =======>                |flow(IP1,...)   |
+...............+                                      +----------------+
```

Figure 9. IP prefix/address anchor switching to the new network with the LM and the FM-CP in a centralized control plane whereas the FM-DPs are distributed.

The example call flow in Figure 10 shows that IP1 is assigned to MN when the MN attaches to the AR1 A flow running in MN and needing IP mobility may continue to use the previous IP prefix by moving the anchoring of the IP prefix to the new network. Yet a new flow to be initiated in the new network may simply use a new IP prefix assigned from the new network.

```
 MN                    AR1           AR2                   CPA        CN
  |MN attaches to AR1:  |             |                     |          |
  |acquire MN-ID and profile          |                     |          |
  |--RS---------------->|             |                     |          |
  |<----------RA(IP1)---|             |                     |          |
  |                     |             |              Assign MN:IP1     |
IP addr config          |             |                     |          |
  |                     |             |                     |          |
  |<-Flow(IP1,IPcn,...)-+--------------------------------------------->|
  |                     |             |                     |          |
  |MN detach from AR1   |             |                     |          |
  |MN attach to AR2     |             |                     |          |
  |                     |             |                     |          |
  |--RS------------------------------>|                     |          |
  |                     |             |                     |          |
  |                     |<---------------control messages-->|          |
  |                     |             |                     |          |
  |                     |             |<-control messages-->|          |
  |                     |             |                     |          |
  |                 forwarding table updates <--------------|          |
  |                     |             |                     |          |
  |<--------------RA(IP2,IP1)---------|                     |          |
  |                     |             |              Assign MN:IP2     |
IP addr config          |             |                     |          |
  |                     |             |                     |          |
  |<-Flow(IP1,IPcn,...)---------------+------------------------------->|
  |                     |             |                     |          |
  |  Flow(IP1,IPcn,...) terminates    |                     |          |
  |                     |             |                     |          |
  |                 forwarding table updates <--------------|          |
  |                     |             |                     |          |
  |<-new Flow(IP2,IPcn,...)-----------+------------------------------->|
  |                     |             |                     |          |
```

Figure 10. DMM solution. MN with flow using IP1 in Net1 continues to run the flow using IP1 as it moves to Net2.

As the MN moves from AR1 to AR2, the AR1 may exchange messages with CPA to release the IP1. It is now necessary for AR2 to learn the IP prefix of the MN from the previous network so that it will be possible for Net2 to assign both the previous network prefix and the new network prefix. The network may learn the previous prefix in different methods. For example, the MN may provide its previous network prefix information by including it to the RS message [I-D.jhlee-dmm-dnpp].

Then forwarding tables updates will take place here.

In addition, the MN also needs a new IP in the new network. The AR2 may now send RA to the MN with prefix information that includes IP1 and IP2. The MN may then continue to use IP1. In addition, the prefix IP2 is assigned to the MN which may configure the IP addresses of its interface. Now for flows using IP1, packets destined to IP1 will be forwarded to the MN via AR2.

As such flows have terminated, IP1 goes back to Net1. MN will then be left with IP2 only, which it will use when it now starts a new flow.

5.2.1. Additional Guidelines for IPv6 Nodes: Switching Anchor with Centralized CP

The configuration guideline for a flat network with centralized control plane and supporting a mix of flows both requiring and not requiring IP mobility support is:

GL-cfg:4 Multiple instances of DPAs (at access routers) which are providing IP prefix to the MNs are needed to provide distributed mobility anchoring according to Figure 1 in Section 3.1 with centralized control plane for a flat network.

At the appropriate IPv6 nodes (CPA, DPA) have to implement the mobility functions LM and FM as described respectively in LM-cfg:1 or LM-cfg:2 and FM-cfg:1 in Section 3.2.

The guidelines (GL-mix) in Section 4.1.1 and in Section 4.2.1 for the IPv6 nodes for a network supporting a mix of flows both requiring and not requiring IP mobility support apply here. The guidelines (GL-mix) in Section 5.1.1 for moving anchoring for a flat network also apply here. In addition, the following are required.

GL-switch:5 It was already mentioned that the anchor operations to properly forward the packets for a flow are described in the FM operations and mobility message parameters in FM-path, FM-path-tbl, FM-cpdp, and FM-DPA in Section 3.2.2 and such changes are reverted later when the application has already closed. Here however, with separation of control and data planes for the anchors and where the LMs and the FM-CP are centralized in the same control plane, messaging between anchors and the discovery of anchors become internal to the control plane.

GL-switch:6 The centralized FM-CP needs to communicate with the distributed FM-DP using the FM operations and mobility message parameters as described in FM-cpdp in Section 3.2.2. Such may be realized by the appropriate messages in [I-D.ietf-dmm-fpc-cpdp].

GL-switch:7 It was also already mentioned before that, if there are in-flight packets toward the previous anchor while the MN is moving to the new anchor, it may be necessary to buffer these packets and then forward to the new anchor after the old anchor knows that the new anchor is ready. Here however, the corresponding FM operations and mobility message parameters as described in Section 3.2.2 (FM-buffer) can be realized by the internal operations in the control plane together with signaling between the control plane and distributed data plane. These signaling may be realized by the appropriate messages in [I-D.ietf-dmm-fpc-cpdp].

5.3. Hierarchical Network

The configuration for a hierarchical network has been shown in Figure 2 in Section 3.1.2. With centralized control plane, CPA and CPN, with the associated LM and FM-CP are all co-located. There are multiple DPAs (each with FM-DP) in distributed mobility anchoring. In the data plane, there are multiple DPNs (each with FM-DP) hierarchically below each DPA. The DPA at each AR supports forwarding to the DPN at each of a number of forwarding switches (FWs). A mobility event in this configuration belonging to distributed mobility management will be deferred to Section 5.4.

In this distributed mobility configuration, a mobility event involving change of FW only but not of AR as shown in Figure 11 may still belong to centralized mobility management and may be supported using PMIPv6. This configuration of network-based mobility is also applicable to host-based mobility with the modification for the MN directly taking the role of DPN and CPN, and the corresponding centralized mobility event may be supported using MIPv6.

In Figure 11, the IP prefix assigned to the MN is anchored at the access router (AR) supporting indirection to the old FW to which the MN was originally attached as well as to the new FW to which the MN has moved.

The realization of LM may be the binding between the IP prefix/ address of the flow used by the MN and the IP address of the DPN to which MN has moved. The implementation of FM to enable change of FW without changing AR may be accomplished using tunneling between the

AR and the FW as described in [I-D.korhonen-dmm-local-prefix] and in [I-D.templin-aerolink] or using some other L2 mobility mechanism.

```
Net1                                                   Net2
+-----------------------------------------------------------------------+
|                    CPA,CPN:  LM:IP1 at IPn2                           |
|                              FM-CP                                    |
+-----------------------------------------------------------------------+

                           +----------------+
                           |AR1             |
                           +----------------+
                           |DPA(IPa1):      |
                           |anchors IP1     |
                           |FM-DP           |
                           +----------------+

+----------------+                                     +----------------+
|FW1             |                                     |FW2             |
+----------------+            FW is changed            +----------------+
|DPN(IPn1):      |               ------->              |DPN(IPn2):      |
|FM-DP           |                                     |FM-DP           |
+----------------+                                     +----------------+

+................+                                     +----------------+
.MN(IP1)         .               MN moves              |MN(IP2)         |
.flow(IP1,...)   .               =======>              |flow(IP1,...)   |
+................+                                     +----------------+
```

Figure 11. Mobility without involving change of IP anchoring in a network in which the IP prefix assigned to the MN is anchored at an AR which is hierarchically above multiple FWs to which the MN may connect.

5.3.1. Additional Guidelines for IPv6 Nodes: Hierarchical Network with No Anchor Relocation

The configuration guideline for a hierarchical network with centralized control plane and supporting a mix of flows both requiring and not requiring IP mobility support is:

GL-cfg:5 Multiple instances of DPAs (at access routers) which are providing IP prefix to the MNs are needed to provide distributed mobility anchoring according to Figure 2 in Section 3.1.2 with centralized control plane for a hierarchical network.

The appropriate IPv6 nodes (CPA, DPA) have to implement the mobility functions LM and FM as described respectively in LM-cfg:3 or LM-cfg:4 and FM-cfg:2 in Section 3.2.

Even when the mobility event does not involve change of anchor, it is still necessary to distinguish whether a flow needs IP mobility support.

The GL-mix guidelines in Section 4.1.1 and in Section 4.2.1 for the IPv6 nodes for a network supporting a mix of flows both requiring and not requiring IP mobility support apply here. In addition, the following are required.

GL-switch:8 Here, the LM operations and mobility message parameters described in Section 3.2.1 provide information of which IP prefix from its FW needs to be used by a flow using which new FW. The anchor operations to properly forward the packets of a flow described in the FM operations and mobility message parameters (FM-path, FM-path-ind, FM-cpdp in Section 3.2.2) may be realized with PMIPv6 protocol [I-D.korhonen-dmm-local-prefix] or with AERO protocol [I-D.templin-aerolink] to tunnel between the AR and the FW.

5.4. IP Prefix/Address Anchor Switching for a Hierarchical Network

The configuration for the hierarchical network has been shown in Figure 2 in Section 3.1.2. Again, with centralized control plane, CPA and CPN, with the associated LM and FM-CP are all co-located. There are multiple DPAs (each with FM-DP) in distributed mobility anchoring. In the data plane, there are multiple DPNs (each with FM-DP) hierarchically below each DPA. The DPA at each AR supports forwarding to the DPN at each of a number of forwarding switches (FWs).

A distributed mobility event in this configuration involves change from a previous DPN which is hierarchically under the previous DPA to a new DPN which is hierarchically under a new DPA. Such an event involving change of both DPA and DPN is shown in Figure 12.

```
Net1                                                   Net2
+----------------------------------------------------------------------+
|       CPA,CPN,Aggregate Router: LM:IP1 at IPn2 at IPa2               |
|                                 FM-CP                                |
+----------------------------------------------------------------------+

                           +-----------------+
                           |Aggregate Router |
                           +-----------------+
                           |FM-DP            |
                           +-----------------+

+---------------+                                      +---------------+
|AR1            |                                      |AR2            |
+---------------+                                      +---------------+
|DPA(IPa1):     | anchoring of IP1 is effectively moved|DPA(IPa2):     |
|anchored IP1   |               =======>               |anchors IP2,IP1|
+---------------+                                      +---------------+

+---------------+                                      +---------------+
|FW1            |                                      |FW2            |
+---------------+             FW is changed            +---------------+
|DPN(IPn1):     |               ------->               |DPN(IPn2):     |
|FM-DP          |                                      |FM-DP          |
+---------------+                                      +---------------+

+...............+                                      +---------------+
.MN(IP1)        .               MN moves               |MN(IP2,IP1)    |
.flow(IP1,...)  .               =======>               |flow(IP1,...)  |
+...............+                                      +---------------+
```

Figure 12. Mobility involving change of IP anchoring in a network with hierarchy in which the IP prefix assigned to the MN is anchored at an Edge Router supporting multiple access routers to which the MN may connect.

This deployment case involves both a change of anchor from AR1 to AR2 and a network hierarchy AR-FW. It can be realized by a combination of relocating the IP prefix/address anchoring from AR1 to AR2 with the mechanism as described in Section 5.2 and then forwarding the packets with network hierarchy AR-FW as described in Section 5.3.

5.4.1. Additional Guidelines for IPv6 Nodes: Switching Anchor with Hierarchical Network

The configuration guideline (GL-cfg) for a hierarchical network with centralized control plane described in Section 5.3.1 applies here.

The GL-mix guidelines in Section 4.1.1 and in Section 4.2.1 for the IPv6 nodes for a network supporting a mix of flows both requiring and not requiring IP mobility support apply here.

The guidelines (GL-switch) in Section 5.1.1 for anchoring relocation and in Section 5.2.1 for a centralized control plane also apply here.

In addition, the guidelines for indirection between the new DPA and the new DPN as described in Section 5.3.1 apply as well.

5.5. Network Mobility

The configuration for network mobility has been shown in Figure 4 in Section 3.1.4. Again, with centralized control plane, CPA, with the associated LM and FM-CP are all co-located. There are multiple DPAs (each with FM-DP) in the data plane in distributed mobility anchoring. The MR possesses the mobility functions FM and LMc. The IP prefix IPn1 is delegated to the MR, to which an MNN is attached and has an IP address from IPn1 assigned to its interface.

Figure 13 shows a distributed mobility event in a hierarchical network with a centralized control plane involving a change of attachment of the MR from a previous DPA to a new DPA while the MNN is attached to the MR and therefore moves with the MR.

```
Net1                                                   Net2
+----------------------------------------------------------------------+
|             CPA,Aggregate Router: LM:IP1 at IPa2; IPn1 at IP1        |
|                                   FM-CP,     LM                      |
+----------------------------------------------------------------------+

                           +-----------------+
                           |Aggregate Router |
                           +-----------------+
                           |FM-DP            |
                           +-----------------+

+---------------+                                      +---------------+
|AR1            |                                      |AR2            |
+---------------+                                      +---------------+
|DPA(IPa1):     | anchoring of IP1 is effectively moved|DPA(IPa2):     |
|anchored IP1   |               =======>               |anchors IP2,IP1|
|DHCPv6-PD IPn1 |                                      |               |
|FM-DP          |                                      |FM-DP          |
+---------------+                                      +---------------+

+...............+                                      +---------------+
.MR(IP1)        .               MR moves               |MR(IP2,IP1)    |
+...............+               =======>               +---------------+
.FM,        LMc .                                      |FM,        LMc |
.delegated IPn1 .                                      |delegated IPn1 |
+...............+                                      +---------------+

+...............+                                      +---------------+
.MNN(IPn1)      .           MNN moves with MR          |MNN(IPn1)      |
.flow(IPn1,...) .               =======>               |flow(IPn1,...) |
+...............+                                      +---------------+
```

Figure 13. Mobility involving change of IP anchoring for a MR to which an MNN is attached.

As the MR with source IP prefix IP1 moves from AR1 to AR2, mobility support may be provided by moving the anchoring of IP1 from AR1 to AR2 using the mechanism described in Section 5.2.

The forwarding table updates will take place at AR1, AR2, the aggregate router, and other affected routers such that the packet from the CN to the MNN will traverse from the aggregate router towards AR2 instead of towards AR1.

5.5.1. Additional Guidelines for IPv6 Nodes: Network mobility

The configuration guideline for a network with centralized control plane to provide network mobility is:

GL-cfg:6 Multiple instances of DPAs (at access routers) which are providing IP prefix of the MRs are needed to provide distributed mobility anchoring according to Figure 4 in Section 3.1.

The appropriate IPv6 nodes (CPA, DPA) have to implement the mobility functions LM and FM as described respectively in LM-cfg:3 or LM-cfg:4 and FM-cfg:4 in Section 3.2.

The GL-mix guidelines in Section 4.1.1 and in Section 4.2.1 for the IPv6 nodes for a network supporting a mix of flows both requiring and not requiring IP mobility support apply here.

Here, because the MN is a MR, the following guideline is added:

GL-mix:11 There are no flows requiring network mobility support when there are no MNN attaching to the MR. Here there are also no MNN using a prefix delegated to the MR. Therefore the anchor of the MR may change to a new AR. The new AR may delegate new IP prefix to the MR, so that the MR may support potential MNNs to attach to it. On the other hand the delegation of IP prefix to the MR from the old AR may be deleted.

The guidelines (GL-switch) in Section 5.1.1 for anchoring relocation and in Section 5.2.1 for a centralized control plane also apply here.

Again because the MN is a MR, the following guidelines are added:

GL-switch:9 Network mobility may be provided using the FM operations and mobility message parameters as described in FM-mr in Section 3.2.2.

GL-switch:10 The following changes to forwarding table entries are needed:

New entries to the forwarding tables are added at AR2 and the aggregate router as well as other affected switches/routers between them so that packets from the CN to the MNN destined to IPn1 will traverse towards AR2. Meanwhile, changes to the forwarding table will also occur at AR1 and the aggregate router as well as other affected switches/routers between them so that in case such packets ever reach any of these switches/ routers, the packets will not traverse towards AR1 but will traverse towards AR2.

GL-switch:11 The security policy must allow the MNN to continue to own the IP prefix/address originally delegated to the MR and used by the MNN at the prior network. As this original IP prefix/address is to be used in the new network, the security policy must allow the anchor node to advertise the prefix of the original IP address and also allow the MNN to send and receive data packets with the original IP address.

GL-switch:12 The security policy must allow the mobile router to configure the original IP prefix/address delegated to the MR from the previous (original) network when the original IP prefix/address is being delegated to the MR in the new network. The security policy must also allows to use the original IP address by the MNNs for the previous flow in the new network.

6. Security Considerations

Security protocols and mechanisms are employed to secure the network and to make continuous security improvements, and a DMM solution is required to support them [RFC7333]. In a DMM deployment [I-D.ietf-dmm-deployment-models] various attacks such as impersonation, denial of service, man-in-the-middle attacks need to be prevented. An appropriate security management function as defined in Section 2 controls these security protocols and mechanisms to provide access control, integrity, authentication, authorization, confidentiality, etc.

Security considerations are described in terms of integrity support, privacy support etc. in describing the mobility functions in Section 3.2. Here the mobility message parameters used in DMM must be protected, and some parameters require means to support MN and MR privacy. The security considerations are also described in the guidelines for IPv6 nodes in various subsections in Section 4, and Section 5.

The IP address anchoring of an IP prefix is effectively moved from one network to another network to support IP mobility Section 5.1. As is considered in the guidelines for IPv6 nodes in Section 5.1.1, the security policy needs to enable the use in the new network of attachment the IP prefix assigned from another network. Yet it must do so without compromising on the needed security to prevent the possible misuse of an IP prefix belonging to another network. A viable solution is likely not be a global solution, but is limited in scope to within specific regions with the proper trust relationship.

In network mobility, the MNN using an IP prefix assigned to it from the MR when the MR was in a prior network moves with the MR to a new network Section 5.5. As is considered in the guidelines for IPv6 nodes in Section 5.5.1 to support IP mobility for an ongoing flow, the security management function needs to enable the continued use of this IP prefix by the MNN with MR in the new network of attachment. Yet it must do so without compromising on the needed security to prevent the possible misuse of an IP prefix belonging to another network. Again, a viable solution is likely not be a global solution, but is limited in scope to within specific regions with the proper trust relationship.

7. IANA Considerations

This document presents no IANA considerations.

8. Contributors

This document has benefited from other work on mobility support in SDN network, on providing mobility support only when needed, and on mobility support in enterprise network. These works have been referenced. While some of these authors have taken the work to jointly write this document, others have contributed at least indirectly by writing these drafts. The latter include Philippe Bertin, Dapeng Liu, Satoru Matushima, Pierrick Seite, Jouni Korhonen, and Sri Gundavelli.

Valuable comments have been received from John Kaippallimalil, ChunShan Xiong, and Dapeng Liu. Dirk von Hugo, Byju Pularikkal, Pierrick Seite, Carlos Bernardos have generously provided careful review with helpful corrections and suggestions.

9. References

9.1. Normative References

[I-D.bernardos-dmm-cmip]
Bernardos, C., Oliva, A., and F. Giust, "An IPv6 Distributed Client Mobility Management approach using existing mechanisms", draft-bernardos-dmm-cmip-07 (work in progress), March 2017.

[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-deployment-models]
Gundavelli, S. and S. Jeon, "DMM Deployment Models and Architectural Considerations", draft-ietf-dmm-deployment-models-01 (work in progress), February 2017.

[I-D.ietf-dmm-fpc-cpdp]
Matsushima, S., Bertz, L., Liebsch, M., Gundavelli, S., Moses, D., and C. Perkins, "Protocol for Forwarding Policy Configuration (FPC) in DMM", draft-ietf-dmm-fpc-cpdp-07 (work in progress), March 2017.

[I-D.ietf-dmm-ondemand-mobility]
Yegin, A., Moses, D., Kweon, K., Lee, J., Park, J., and S. Jeon, "On Demand Mobility Management", draft-ietf-dmm-ondemand-mobility-11 (work in progress), June 2017.

[I-D.jhlee-dmm-dnpp]
Lee, J. and Z. Yan, "Deprecated Network Prefix Provision", draft-jhlee-dmm-dnpp-01 (work in progress), April 2016.

[I-D.korhonen-dmm-local-prefix]
Korhonen, J., Savolainen, T., and S. Gundavelli, "Local Prefix Lifetime Management for Proxy Mobile IPv6", draft-korhonen-dmm-local-prefix-01 (work in progress), July 2013.

[I-D.liu-dmm-deployment-scenario]
Liu, V., Liu, D., Chan, A., Lingli, D., and X. Wei, "Distributed mobility management deployment scenario and architecture", draft-liu-dmm-deployment-scenario-05 (work in progress), October 2015.

[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.

[I-D.mccann-dmm-prefixcost]
McCann, P. and J. Kaippallimalil, "Communicating Prefix Cost to Mobile Nodes", draft-mccann-dmm-prefixcost-03 (work in progress), April 2016.

[I-D.sarikaya-dmm-for-wifi]
Sarikaya, B. and L. Xue, "Distributed Mobility Management Protocol for WiFi Users in Fixed Network", draft-sarikaya-dmm-for-wifi-04 (work in progress), March 2016.

[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.templin-aerolink]
Templin, F., "Asymmetric Extended Route Optimization (AERO)", draft-templin-aerolink-75 (work in progress), May 2017.

[I-D.yhkim-dmm-enhanced-anchoring]
Kim, Y. and S. Jeon, "Enhanced Mobility Anchoring in Distributed Mobility Management", draft-yhkim-dmm-enhanced-anchoring-05 (work in progress), July 2016.

[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>.

[RFC3753] Manner, J., Ed. and M. Kojo, Ed., "Mobility Related Terminology", RFC 3753, DOI 10.17487/RFC3753, June 2004, <http://www.rfc-editor.org/info/rfc3753>.

[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>.

[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>.

[RFC7077] Krishnan, S., Gundavelli, S., Liebsch, M., Yokota, H., and J. Korhonen, "Update Notifications for Proxy Mobile IPv6", RFC 7077, DOI 10.17487/RFC7077, November 2013, <http://www.rfc-editor.org/info/rfc7077>.

[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>.

9.2. Informative References

[Paper-Distributed.Mobility]
Lee, J., Bonnin, J., Seite, P., and H. Chan, "Distributed IP Mobility Management from the Perspective of the IETF: Motivations, Requirements, Approaches, Comparison, and Challenges", IEEE Wireless Communications, October 2013.

[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 (editor)
Huawei Technologies
5340 Legacy Dr. Building 3
Plano, TX 75024
USA

Email: h.a.chan@ieee.org

Xinpeng Wei
Huawei Technologies
Xin-Xi Rd. No. 3, Haidian District
Beijing, 100095
P. R. China

Email: weixinpeng@huawei.com

Jong-Hyouk Lee
Sangmyung University
31, Sangmyeongdae-gil, Dongnam-gu
Cheonan 31066
Republic of Korea

Email: jonghyouk@smu.ac.kr

Seil Jeon
Sungkyunkwan University
2066 Seobu-ro, Jangan-gu
Suwon, Gyeonggi-do
Republic of Korea

Email: seiljeon@skku.edu

Alexandre Petrescu
CEA, LIST
CEA Saclay
Gif-sur-Yvette, Ile-de-France 91190
France

Phone: +33169089223
Email: Alexandre.Petrescu@cea.fr

Fred L. Templin
Boeing Research and Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 17, 2018

S. Matsushima
SoftBank
L. Bertz
Sprint
M. Liebsch
NEC
S. Gundavelli
Cisco
D. Moses
Intel Corporation
C. Perkins
Futurewei

September 13, 2017

Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-08

Abstract

This document describes a way, called Forwarding Policy Configuration (FPC) to manage the separation of data-plane and control-plane. FPC defines a flexible mobility management system using FPC agent and FPC client functions. An FPC agent provides an abstract interface to the data-plane. The FPC client configures data-plane nodes by using the functions and abstractions provided by the FPC agent for that data-plane nodes. The data-plane abstractions presented in this document is extensible, in order to support many different types of mobility management systems and data-plane functions.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes Forwarding Policy Configuration (FPC), a system for managing the separation of data-plane and control-plane. FPC enables flexible mobility management using FPC agent and FPC client functions. An FPC agent exports an abstract interface to the data-plane. To configure data-plane nodes and functions, the FPC client uses the interface to the data-plane offered by the FPC agent.

Control planes of mobility management systems, or other applications which require data-plane control, can utilize the FPC client at various granularities of operation. The operations are capable of configuring a single Data-Plane Node (DPN) directly, as well as multiple DPNs as determined by abstracted data-plane models on the FPC agent.

A FPC agent provides data-plane abstraction in the following three areas:

Topology: DPNs are grouped and abstracted according to well-known concepts of mobility management such as access networks, anchors and domains. A FPC agent provides an interface to the abstract DPN-groups that enables definition of a topology for the forwarding plane. For example, access nodes may be assigned to a DPN-group which peers to a DPN-group of anchor nodes.

Policy: A Policy embodies the mechanisms for processing specific traffic flows or packets. This is needed for QoS, for packet processing to rewrite headers, etc. A Policy consists of one or more rules. Each rule is composed of Descriptors and Actions. Descriptors in a rule identify traffic flows, and Actions apply treatments to packets that match the Descriptors in the rule. An arbitrary set of policies can be abstracted as a Policy-group to be applied to a particular collection of flows, which is called the Virtual Port (Vport).

Mobility: A mobility session which is active on a mobile node is abstracted as a Context with associated runtime concrete attributes, such as tunnel endpoints, tunnel identifiers, delegated prefix(es), routing information, etc. Contexts are attached to DPN-groups along with consequence of the control plane. One or multiple Contexts which have same sets of policies are assigned Vports which abstract those policy sets. A Context can belong to multiple Vports which serve various kinds of purpose and policy. Monitors provide a mechanism to produce reports when events regarding Vports, Sessions, DPNs or the Agent occur.

The Agent assembles applicable sets of forwarding policies for the mobility sessions from the data model, and then renders those policies into specific configurations for each DPN to which the sessions attached. The specific protocols and configurations to configure DPN from a FPC Agent are outside the scope of this document.

The data-plane abstractions may be extended to support many different mobility management systems and data-plane functions. The architecture and protocol design of FPC is not tied to specific types of access technologies and mobility protocols.

2. 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].

DPN:	A data-plane node (DPN) is capable of deploying data-plane features. DPNs may be

	switches or routers regardless of their realiziation, i.e. whether they are hardware or software based.
FPC Agent:	A functional entity in FPC that manages DPNs and provides abstracted data-plane networks to mobility management systems and/or applications through FPC Clients.
FPC Client:	A functional entity in FPC that is integrated with mobility management systems and/or applications to control forwarding policy, mobility sessions and DPNs.
Tenant:	An operational entity that manages mobility management systems or applications which require data-plane functions.
Domain:	One or more DPNs that form a data-plane network. A mobility management system or an application in a tenant may utilize a single or multiple domains.
Virtual Port (Vport):	A set of forwarding policies.
Context:	An abstracted endpoint of a mobility session associated with runtime attributes. Vports may apply to Context which instantiates those forwarding policies on a DPN.

3. FPC Architecture

To fulfill the requirements described in [RFC7333], FPC enables mobility control-planes and applications to configure DPNs with various roles of the mobility management as described in [I-D.ietf-dmm-deployment-models].

FPC defines building blocks of FPC Agent and FPC Client, as well as data models for the necessary data-plane abstractions. The attributes defining those data models serve as protocol elements for the interface between the FPC Agent and the FPC Client.

Mobility control-planes and applications integrate the FPC Client function. The FPC Client connects to FPC Agent functions. The Client and the Agent communicate based on information models for the data-plane abstractions described in Section 4. The data models allow the control-plane and the applications to support forwarding policies on the Agent for their mobility sessions.

The FPC Agent carries out the required configuration and management of the DPN(s). The Agent determines DPN configurations according to the forwarding policies requested by the FPC Client. The DPN configurations could be specific to each DPN implementation such that how FPC Agent determines implementation specific configuration for a DPN is outside of the scope of this document. Along with the models, the control-plane and the applications put Policies to the Agent prior to creating their mobility sessions.

Once the Topology of DPN(s) and domains are defined for a data plane on an Agent, the data-plane nodes (DPNs) are available for further configuration. The FPC Agent connects those DPNs to manage their configurations.

This architecture is illustrated in Figure 1. An FPC Agent may be implemented in a network controller that handles multiple DPNs, or there is a simple case where another FPC Agent may itself be integrated into a DPN.

This document does not adopt a specific protocol for the FPC interface protocol and it is out of scope. However it must be capable of supporting FPC protocol messages and transactions described in Section 5.

```
                           +-------------------------+
                           | Mobility Control-Plane  |
                           |          and            |
                           |      Applications       |
                           |+-----------------------+|
                           ||      FPC Client       ||
                           |+----------^------------+|
                           +-----------|-------------+
               FPC interface protocol  |
                       +---------------+-----------------+
                       |                                 |
         Network       |                                 |
         Controller    |                      DPN        |
           +-----------|-------------+        +----------|---------+
           |+----------v------------+|        |+---------v--------+|
           ||   [Data-plane model]  ||        ||[Data-plane model]||
           ||       FPC Agent       ||        ||    FPC Agent     ||
           |+-----------------------+|        |+------------------+|
           |+------------+----------+|        |                    |
           ||SB Protocols|FPC Client||        |  DPN Configuration |
           ||   Modules  |  Module  ||        +--------------------+
           |+------^-----+----^-----+|
           +-------|----------|------+
                   |          |
         Other     |          | FPC interface
         Southband |          | Protocol
         Protocols |          |
                   |          +-----------------+
                   |                            |
       DPN         |                 DPN        |
        +----------|---------+       +----------|---------+
        |+---------v--------+|       |+---------v--------+|
        ||  Configuration   ||       ||[Data-plane model]||
        || Protocol module  ||       ||     FPC Agent    ||
        |+------------------+|       |+------------------+|
        |                    |       |                    |
        | DPN Configuration  |       |  DPN Configuration |
        +--------------------+       +--------------------+
```

Figure 1: Reference Forwarding Policy Configuration (FPC) Architecture

The FPC architecture supports multi-tenancy; an FPC enabled data-plane supports tenants of multiple mobile operator networks and/or applications. It means that the FPC Client of each tenant connects to the FPC Agent and it MUST partition namespace and data for their data-planes. DPNs on the data-plane may fulfill multiple data-plane roles which are defined per session, domain and tenant.

Note that all FPC models SHOULD be configurable. The FPC interface protocol in Figure 1 is only required to handle runtime data in the Mobility model. The rest of the FPC models, namely Topology and Policy, may be pre-configured, and in that case real-time protocol exchanges would not be required for them. Operators that are tenants in the FPC data-plane could configure Topology and Policy on the Agent through other means, such as Restconf [I-D.ietf-netconf-restconf] or Netconf [RFC6241].

4. Information Model for FPC

This section presents an information model representing the abstract concepts of FPC, which are language and protocol neutral. Figure 2 shows an overview of the FPC data-plane information model.

```
     (Mobile operator tenant that abstracted data-plane is used)
             |
             +---FPC-Topology
             |      |
             |      +---DPNs
             |      |
             |      +---DPN-groups
             |      |
             |      +---Domains
             |
             +---FPC-Policy
             |     |
             |     +---Descriptors
             |     |
             |     +---Actions
             |     |
             |     +---Policies
             |     |
             |     +---Policy-groups
             |
             +---FPC-Mobility
                    |
                    +---Vports
                    |
                    +---Contexts
```

Figure 2: FPC Data-plane Information Model

4.1. FPC-Topology

Topology abstraction enables a physical data-plane network to support multiple overlay topologies. An FPC-Topology consists of DPNs, DPN-groups and Domains which abstract data-plane topologies for the Client’s mobility control-planes and applications.

Utilizing a FPC Agent, a mobile operator can create virtual DPNs in an overlay network. Those such virtual DPNs are treated the same as physical forwarding DPNs in this document.

4.1.1. DPNs

The DPNs define all available nodes to a tenant of the FPC data-plane network. FPC Agent defines DPN binding to actual nodes. The role of a DPN in the data-plane is determined at the time the DPN is assigned to a DPN-group.

```
                  (FPC-Topology)
                      |
                      +---DPNs
                             |
                             +---DPN-id
                             |
                             +---DPN-name
                             |
                             +---DPN-groups
                             |
                             +---Node-reference
```

Figure 3: DPNs Model Structure

DPN-id: The identifier for the DPN. The ID format MUST conform to Section 4.4.

DPN-name: The name of the DPN.

DPN-groups: The list of DPN-groups to which the DPN belongs.

Node-reference: Indicates a physical node, or a platform of virtualization, to which the DPN is bound by the Agent. The Agent SHOULD maintain that node’s information, including IP address of management and control protocol to connect them. In the case of a node as a virtualization platform, FPC Agent directs the platform to instantiate a DPN to which a DPN-group attributes.

4.1.2. DPN-groups

A DPN-group is a set of DPNs which share certain specified data-plane attributes. DPN-groups define the data-plane topology consisting of a DPN-group of access nodes connecting to an anchor node’s DPN-group.

A DPN-group has attributes such as its data-plane role, supported access technologies, mobility profiles, connected peer groups and domain. A DPN may be assigned to multiple DPN-groups in different data-plane roles or different domains.

```
                  (FPC-Topology)
                      |
                      +---DPN-groups
                             |
                             +---DPN-group-id
                             |
                             +---Data-plane-role
                             |
                             +---Domains
                             |
                             +---Access-type
                             |
                             +---Mobility-profile
                             |
                             +---DPN-group-peers
```

Figure 4: DPN-groups Model Structure

DPN-group-id: The identifier of the DPN-group. The ID format MUST conform to Section 4.4.

Data-plane-role: The data-plane role of the DPN-group, such as access-dpn, anchor-dpn.

Domains: The domains to which the DPN-group belongs.

Access-type: The access type supported by the DPN-group such as ethernet(802.3/11), 3gpp cellular(S1, RAB), if any.

Mobility-profile: Identifies a supported mobility profile, such as ietf-pmip, or 3gpp. New profiles may be defined as extensions of this specification. Mobility profiles are defined so that some or all data-plane parameters of the mobility contexts that are part of the profile can be automatically determined by the FPC Agent.

DPN-group-peers: The remote peers of the DPN-group with parameters described in Section 4.1.2.1.

4.1.2.1. DPN-group Peers

DPN-group-peers lists relevant parameters of remote peer DPNs as illustrated in Figure 5.

```
                   (DPN-groups)
                       |
                       +---DPN-group-peers
                             |
                             +---Remote-DPN-group-id
                             |
                             +---Remote-mobility-profile
                             |
                             +---Remote-data-plane-role
                             |
                             +---Remote-endpoint-address
                             |
                             +---Local-endpoint-address
                             |
                             +---MTU-size
```

Figure 5: DPN-groups Peer Model Structure

Remote-DPN-group-id: The ID of the peering DPN-Group. The ID format MUST conform to Section 4.4.

Remote-mobility-profile: The mobility-profile for the peering DPN-group. Currently defined profiles are ietf-pmip, or 3gpp. New profiles may be defined as extensions of this specification.

Remote-data-plane-role: The data-plane role of the peering DPN-group.

Remote-endpoint-address: Defines Endpoint address of the peering DPN-group.

Local-endpoint-address: Defines Endpoint address of its own DPN-group to peer the remote DPN-group.

MTU-size: Defines MTU size of traffic between the DPN-Group and this DPN-group-peer.

4.1.3. Domains

A domain is defined by an operator to refer to a particular network, considered as a system of cooperating DPN-groups. Domains may represent services or applications that are resident within an operator’s network.

```
                      (FPC-Topology)
                          |
                          +---Domains
                                 |
                                 +---Domain-id
                                 |
                                 +---Domain-name
                                 |
                                 +---Domain-type
                                 |
                                 +---Domain-reference
```

Figure 6: Domain Model Structure

Domain-id: Identifier of Domain. The ID format MUST conform to Section 4.4.

Domain-name: The name of the Domain.

Domain-type: Specifies which address families are supported within the domain.

Domain-reference: Indicates a set of resources for the domain which consists a topology of physical nodes, platforms of virtualization and physical/virtual links with certain bandwidth, etc,.

4.2. FPC-Policy

The FPC-Policy consists of Descriptors, Actions, Policies and Policy-groups. These can be viewed as configuration data, in contrast to Contexts and Vports, which are structures that are instantiated on the Agent. The Descriptors and Actions in a Policy referenced by a Vport are active when the Vport is in an active Context, i.e. they can be applied to traffic on a DPN.

4.2.1. Descriptors

Descriptors defines classifiers of specific traffic flows, such as those based on source and destination addresses, protocols, port numbers of TCP/UDP/SCTP/DCCP, or any way of classifying packets. Descriptors are defined by specific profiles that may be produced by 3gpp, ietf or other SDOs. Many specifications also use the terms Filter, Traffic Descriptor or Traffic Selector [RFC6088]. A packet that meets the criteria of a Descriptor is said to satisfy, pass or be consumed by the Descriptor. Descriptors are assigned an identifier and contain a type and value.

```
                  (FPC-Policy)
                      |
                      +---Descriptors
                             |
                             +---Descriptor-id
                             |
                             +---Descriptor-type
                             |
                             +---Descriptor-value
```

Figure 7: Descriptor Model Structure

Descriptor-id: Identifier of Descriptor. The ID format MUST conform to Section 4.4.

Descriptor-type: The descriptor type, which determines the classification of a specific traffic flows, such as source and destination addresses, protocols, port numbers of TCP/UDP/SCTP/ DCCP, or any other way of selecting packets.

Descriptor-value: The value of Descriptor such as IP prefix/address, protocol number, port number, etc.

4.2.2. Actions

A Policy defines a list of Actions that are to be applied to traffic meeting the criteria defined by the Descriptors. Actions include traffic management such as shaping, policing based on given bandwidth, and connectivity actions such as pass, drop, forward to given nexthop. Actions may be defined as part of specific profiles which are produced by 3gpp, ietf or other SDOs.

```
                        (FPC-Policy)
                            |
                            +---Actions
                                   |
                                   +---Action-id
                                   |
                                   +---Action-type
                                   |
                                   +---Action-value
```

Figure 8: Action Model Structure

Action-id: Identifier for the Action. The ID format MUST conform to Section 4.4.

Action-type: The type of the action -- i.e. how to treat the specified traffic flows. Examples include pass, drop, forward to a given nexthop value, shape or police based on given bandwidth value, etc.

Action-value: Specifies a value for the Action-type, such as bandwidth, nexthop address or drop, etc.

4.2.3. Policies

Policies are collections of Rules. Each Policy has a Policy Identifier and a list of Rule/Order pairs. The Order and Rule values MUST be unique in the Policy. Unlike the AND filter matching of each Rule the Policy uses an OR matching to find the first Rule whose Descriptors are satisfied by the packet. The search for a Rule to apply to packet is executed according to the unique Order values of the Rules. This is an ascending order search, i.e. the Rule with the lowest Order value is tested first and if its Descriptors are not satisfied by the packet the Rule with the next lowest Order value is tested. If a Rule is not found then the Policy does not apply. Policies contain Rules (not references to Rules).

```
                 (FPC-Policy)
                     |
                     +---Policies
                            |
                            +---Policy-id
                            |
                            +---Rules
                                  |
                                  +---Order
                                  |
                                  +---Descriptors
                                  |      |
                                  |      +---Descriptor-id
                                  |      |
                                  |      +---Direction
                                  |
                                  +---Actions
                                         |
                                         +---Action-id
                                         |
                                         +---Action-Order
```

Figure 9: Model Structure for Policies

Policy-id: Identifier of Policy. The ID format MUST conform to Section 4.4.

Rules: List of Rules which are a collection of Descriptors and Actions. All Descriptors MUST be satisfied before the Actions are taken. This is known as an AND Descriptor list, i.e. Descriptor 1 AND Descriptor 2 AND ... Descriptor X all MUST be satisfied for the Rule to apply.

Order: Specifies ordering if the Rule has multiple Descriptors and Action sets. Order values MUST be unique within the Rules list.

Descriptors: The list of Descriptors.

Descriptor-id: Identifies each Descriptor in the Rule.

Direction: Specifies which direction applies, such as uplink, downlink or both.

Actions: List of Actions.

Action-id: Indicates each Action in the rule.

Action-Order: Specifies Action ordering if the Rule has multiple actions. Action-Order values MUST be unique within the Actions list.

4.2.4. Policy-groups

List of Policy-groups which are an aggregation of Policies. Common applications include aggregating Policies that are defined by different functions, e.g. Network Address Translation, Security, etc. The structure has an Identifier and references the Policies via their Identifiers.

```
                      (FPC-Policy)
                          |
                          +---Policy-groups
                                 |
                                 +---Policy-group-id
                                 |
                                 +---Policies
```

Figure 10: Policy-group Model Structure

Policy-group-id: The identifier of the Policy-group. The ID format MUST conform to Section 4.4.

Policies: List of Policies in the Policy-group.

4.3. FPC for Mobility Management

The FPC-Mobility consists of Vports and Contexts. A mobility session is abstracted as a Context with its associated runtime concrete attributes, such as tunnel endpoints, tunnel identifiers, delegated prefix(es) and routing information, etc. A Vport abstracts a set of policies applied to the Context.

4.3.1. Vport

A Vport represents a collection of policy groups, that is, a group of rules that can exist independently of the mobility/session lifecycle. Mobility control-plane applications create, modify and delete Vports on FPC Agent through the FPC Client.

When a Vport is indicated in a Context, the set of Descriptors and Actions in the Policies of the Vport are collected and applied to the Context. They must be instantiated on the DPN as forwarding related actions such as QoS differentiations, packet processing of encap/ decap, header rewrite, route selection, etc.

```
                     (FPC-Mobility)
                             |
                             +---Vports
                                    |
                                    +---Vport-id
                                    |
                                    +---Policy-groups
```

Figure 11: Vport Model Structure

Vport-id: The identifier of Vport. The ID format MUST conform to Section 4.4.

Policy-groups: List of references to Policy-groups which apply to the Vport.

4.3.2. Context

An endpoint of a mobility session is abstracted as a Context with its associated runtime concrete attributes, such as tunnel endpoints, tunnel identifiers, delegated prefix(es) and routing information, etc. A mobility control-plane, or other applications, can create, modify and delete contexts on an FPC Agent by using the FPC Client.

FPC Agent SHOULD determine runtime attributes of a Context from the Vport’s policies and the attached DPN’s attributes. A mobility control-plane, or other applications, MAY set some of the runtime attributes directly when they create data-plane related attributes. In the case of that a mobility control-plane assigns tunnel identifiers, for instance.

```
                 (FPC-Mobility)
                         |
                         +---Contexts
                                |
                                +---Context-id
                                |
                                +---Vports
                                |
                                +---DPN-group
                                |
                                +---Delegated-ip-prefixes
                                |
                                +---Parent-context
```

Figure 12: Common Context Model Structure

Context-id: Identifier of the Context. The ID format MUST conform to Section 4.4.

Vports: List of Vports. When a Context is applied to a Vport, the context is configured by policies at each such Vport. Vport-id references indicate Vports which apply to the Context. Context can be a spread over multiple Vports which have different policies.

DPN-group: The DPN-group assigned to the Context.

Delegated-ip-prefixes: List of IP prefixes to be delegated to the mobile node of the Context.

Parent-context: Indicates a parent context from which this context inherits.

4.3.2.1. Single DPN Agent Case

In the case where a FPC Agent supports only one DPN, the Agent MUST maintain Context data just for the DPN. The Agent does not need to maintain a Topology model. Contexts in single DPN case consists of following parameters for both direction of uplink and downlink.

```
              (Contexts)
                  |
                  +---UL-Tunnel-local-address
                  |
                  +---UL-Tunnel-remote-address
                  |
                  +---UL-MTU-size
                  |
                  +---UL-Mobility-specific-tunnel-parameters
                  |
                  +---UL-Nexthop
                  |
                  +---UL-QoS-profile-specific-parameters
                  |
                  +---UL-DPN-specific-parameters
                  |
                  +---UL-Vendor-specific-parameters
```

Figure 13: Uplink Context Model of Single DPN Structure

UL-Tunnel-local-address: Specifies uplink endpoint address of the DPN.

UL-Tunnel-remote-address: Specifies uplink endpoint address of the remote DPN.

UL-MTU-size: Specifies the uplink MTU size.

UL-Mobility-specific-tunnel-parameters: Specifies profile specific uplink tunnel parameters to the DPN which the agent exists. This may, for example, include GTP/TEID for 3gpp profile, or GRE/Key for ietf-pmip profile.

UL-Nexthop: Indicates next-hop information of uplink in external network such as IP address, MAC address, SPI of service function chain [I-D.ietf-sfc-nsh], SID of segment routing[I-D.ietf-6man-segment-routing-header] [I-D.ietf-spring-segment-routing-mpls], etc.

UL-QoS-profile-specific-parameters: Specifies profile specific QoS parameters of uplink, such as QCI/TFT for 3gpp profile, [RFC6089]/[RFC7222] for ietf-pmip, or parameters of new profiles defined by extensions of this specification.

UL-DPN-specific-parameters: Specifies optional node specific parameters needed by uplink such as if-index, tunnel-if-number that must be unique in the DPN.

UL-Vendor-specific-parameters: Specifies a vendor specific parameter space for the uplink.

```
              (Contexts)
                  |
                  +---DL-Tunnel-local-address
                  |
                  +---DL-Tunnel-remote-address
                  |
                  +---DL-MTU-size
                  |
                  +---DL-Mobility-specific-tunnel-parameters
                  |
                  +---DL-Nexthop
                  |
                  +---DL-QoS-profile-specific-parameters
                  |
                  +---DL-DPN-specific-parameters
                  |
                  +---DL-Vendor-specific-parameters
```

Figure 14: Downlink Context Model of Single DPN Structure

DL-Tunnel-local-address: Specifies downlink endpoint address of the DPN.

DL-Tunnel-remote-address: Specifies downlink endpoint address of the remote DPN.

DL-MTU-size: Specifies the downlink MTU size of tunnel.

DL-Mobility-specific-tunnel-parameters: Specifies profile specific downlink tunnel parameters to the DPN which the agent exists. This may, for example, include GTP/TEID for 3gpp profile, or GRE/ Key for ietf-pmip profile.

DL-Nexthop: Indicates next-hop information of downlink in external network such as IP address, MAC address, SPI of service function chain [I-D.ietf-sfc-nsh], SID of segment routing[I-D.ietf-6man-segment-routing-header] [I-D.ietf-spring-segment-routing-mpls], etc.

DL-QoS-profile-specific-parameters: Specifies profile specific QoS parameters of downlink, such as QCI/TFT for 3gpp profile, [RFC6089]/[RFC7222] for ietf-pmip, or parameters of new profiles defined by extensions of this specification.

DL-DPN-specific-parameters: Specifies optional node specific parameters needed by downlink such as if-index, tunnel-if-number that must be unique in the DPN.

DL-Vendor-specific-parameters: Specifies a vendor specific parameter space for the downlink.

4.3.2.2. Multiple DPN Agent Case

Alternatively, a FPC Agent may connect to multiple DPNs. The Agent MUST maintain a set of Context data for each DPN. The Context contains a list of DPNs, where each entry of the list consists of the parameters in Figure 15. A Context data for one DPN has two entries - one for uplink and another for downlink or, where applicable, a direction of ’both’.

```
             (Contexts)
                 |
                 +---DPNs
                      |
                      +---DPN-id
                      |
                      +---Direction
                      |
                      +---Tunnel-local-address
                      |
                      +---Tunnel-remote-address
                      |
                      +---MTU-size
                      |
                      +---Mobility-specific-tunnel-parameters
                      |
                      +---Nexthop
                      |
                      +---QoS-profile-specific-parameters
                      |
                      +---DPN-specific-parameters
                      |
                      +---Vendor-specific-parameters
```

Figure 15: Multiple-DPN Supported Context Model Structure

DPN-id: Indicates DPN of which the runtime Context data installed.

Direction: Specifies which side of connection at the DPN indicated - uplink, downlink or both.

Tunnel-local-address: Specifies endpoint address of the DPN at the uplink or downlink.

Tunnel-remote-address: Specifies endpoint address of remote DPN at the uplink or downlink.

MTU-size: Specifies the packet MTU size on uplink or downlink.

Mobility-specific-tunnel-parameters: Specifies profile specific tunnel parameters for uplink or downlink to the DPN. This may, for example, include GTP/TEID for 3gpp profile, or GRE/Key for ietf-pmip profile.

Nexthop: Indicates next-hop information for uplink or downlink in external network such as IP address, MAC address, SPI of service function chain [I-D.ietf-sfc-nsh], SID of segment routing[I-D.ietf-6man-segment-routing-header] [I-D.ietf-spring-segment-routing-mpls], etc.

QoS-profile-specific-parameters: Specifies profile specific QoS parameters for uplink or downlink to the DPN, such as QCI/TFT for 3gpp profile, [RFC6089]/[RFC7222] for ietf-pmip, or parameters of new profiles defined by extensions of this specification.

DPN-specific-parameters: Specifies optional node specific parameters needed by uplink or downlink to the DPN such like if-index, tunnel-if-number that must be unique in the DPN.

Vendor-specific-parameters: Specifies a vendor specific parameter space for the DPN.

Multi-DPN Agents will use only the DPNs list of a Context for processing as described in this section. A single-DPN Agent MAY use both the Single Agent DPN model Section 4.3.2.1 and the multi-DPN Agent Context described here.

4.3.3. Monitors

Monitors provide a mechanism to produce reports when events occur. A Monitor will have a target that specifies what is to be watched.

When a Monitor is specified, the configuration MUST be applicable to the attribute/entity monitored. For example, a Monitor using a Threshold configuration cannot be applied to a Context, because Contexts do not have thresholds. But such a monitor could be applied to a numeric threshold property of a Context.

```
                     (FPC-Mobility)
                             |
                             +---Monitors
                                    |
                                    +---Monitor-id
                                    |
                                    +---Target
                                    |
                                    +---Configuration
```

Figure 16: Common Monitor Model Structure

Monitor-id: Name of the Monitor. The ID format MUST conform to Section 4.4.

Target: Target to be monitored. This may be an event, a Context, a Vport or attribute(s) of Contexts. When the type is an attribute(s) of a Context, the target name is a concatenation of the Context-Id and the relative path (separated by '/') to the attribute(s) to be monitored.

Configuration: Determined by the Monitor subtype. Four report types are defined:

* Periodic reporting specifies an interval by which a notification is sent to the Client.

* Event reporting specifies a list of event types that, if they occur and are related to the monitored attribute, will result in sending a notification to the Client.

* Scheduled reporting specifies the time (in seconds since Jan 1, 1970) when a notification for the monitor should be sent to the Client. Once this Monitor's notification is completed the Monitor is automatically de-registered.

* Threshold reporting specifies one or both of a low and high threshold. When these values are crossed a corresponding notification is sent to the Client.

4.4. Namespace and Format

The identifiers and names in FPC models which reside in the same namespace must be unique. That uniqueness must be kept in agent or data-plane tenant namespace on an Agent. The tenant namespace uniqueness MUST be applied to all elements of the tenant model, i.e. Topology, Policy and Mobility models.

When a Policy needs to be applied to Contexts in all tenants on an Agent, the Agent SHOULD define that policy to be visible from all the tenants. In this case, the Agent assigns an unique identifier in the agent namespace.

The format of identifiers can utilize any format with agreement between data-plane agent and client operators. The formats include but are not limited to Globally Unique IDentifiers (GUIDs), Universally Unique IDentifiers (UUIDs), Fully Qualified Domain Names (FQDNs), Fully Qualified Path Names (FQPNs) and Uniform Resource Identifiers (URIs).

The FPC model does not limit the types of format that dictate the choice of FPC protocol. However the choice of identifiers which are used in Mobility model need to be considered to handle runtime parameters in real-time. The Topology and Policy models are not restricted to meet that requirement, as described in Section 3.

4.5. Attribute Application

Attributes in FPC Topology and Policy SHOULD be pre-configured in a FPC Agent prior to Contexts and Vports. The FPC Agent requires those pre-configured attributes to be able to derive a Context’s detailed runtime attributes.

When a FPC Client creates a Context, the FPC Client is then able to indicate specific DPN-group(s) instead of all endpoint addresses of the DPN(s) and MTU-size of the tunnels for example. This is because that the FPC Agent can derive data for those details from the pre-configured DPN-group information in the FPC Topology.

Similarly when a Vport is created for the Context, the FPC Agent can derive detailed forwarding policies from the pre-configured Policy information in the FPC Policy. The FPC Client thereby has no need to indicate those specific policies to all of the Contexts which share the same set of Policy-groups.

This is intentional as it provides FPC Clients the ability to reuse pre-configured FPC Topology and FPC Policy attributes. It helps to minimize over the wire exchanges and reduce system errors by exchanging less information.

The Agent turns those derived data into runtime attributes of UL and DL objects which are in the DPNs list of the Context (multiple-DPNs Agent case) or directly under the Context (single-DPN Agent case). The Agent consequently instantiates forwarding policies on DPN(s) based on those attributes.

When a Context inherits another Context as its parent, missing attributes in the child Context are provided by the Parent Context (for example, IMSI defined in the 3GPP extension) .

It is noted that the Agent SHOULD update the Context’s attributes which are instantiated on DPN(s) when the applied attributes of Topology and Policy are changed.

In the case of FPC Client modifying an existing runtime attribute of a Context which the FPC Agent derived, the FPC Agent MUST overwrite that attribute with the value which the Client brings to the Agent. However risks exist, for example, the attributes could be outside of allowable range of DPNs which the FPC Agent managed.

5. Protocol

5.1. Protocol Messages and Semantics

Five message types are supported:

Message	Type	Description
CONF	HEADER ADMIN_STATE SESSION_STATE OP_TYPE BODY	Configure processes a single operation.
CONF_BUNDLE	1*[HEADER ADMIN_STATE SESSION_STATE TRANS_STRATEGY OP_TYPE BODY]	A Conf-bundle takes multiple operations that are to be executed as a group with partial failures allowed. They are executed according to the OP_ID value in the OP_BODY in ascending order. If a CONF_BUNDLE fails, any entities provisioned in the CURRENT operation are removed. However, any successful operations completed prior to the current operation are preserved in order to reduce system load.
REG_MONITOR	HEADER ADMIN_STATE *[MONITOR]	Register a monitor at an Agent. The message includes information about the attribute to monitor and the reporting method. Note that a MONITOR_CONFIG is required for this operation.
DEREG_MONITOR	HEADER *[MONITOR_ID] [boolean]	Deregister monitors from an Agent. Monitor IDs are provided. Boolean (optional) indicates if a successful DEREG triggers a NOTIFY with final data.
PROBE	HEADER MONITOR_ID	Probe the status of a registered monitor.

Table 1: Client to Agent Messages

Each message contains a header with the Client Identifier, an execution delay timer and an operation identifier. The delay, in ms, is processed as the delay for operation execution from the time the operation is received by the Agent.

The Client Identifier is used by the Agent to associate specific configuration characteristics, e.g. options used by the Client when communicating with the Agent, as well as the association of the Client and tenant in the information model.

Messages that create or update Monitors and Entities, i.e. CONFIG, CONF_BUNDLE and REG_MONITOR, specify an Administrative State which specifies the Administrative state of the message subject(s) after the successful completion of the operation. If the status is set to virtual, any existing data on the DPN is removed. If the value is set to disabled, and if that entity exists on the DPN, then an operation to disable the associated entity will occur on the DPN . If set to 'active' the DPN will be provisioned. Values are 'enabled', 'disabled', and 'virtual'.

CONF_BUNDLE also has the Transaction Strategy (TRANS_STRATEGY) attribute. This value specifies the behavior of the Agent when an operation fails while processing a CONF_BUNDLE message. The value of 'default' uses the default strategy defined for the message. The value 'all_or_nothing' will roll back all successfully executed operations within the bundle as well as the operation that failed.

An FPC interface protocol used to support this specification may not need to support CONF_BUNDLE messages or specific TRANS_STRATEGY types beyond 'default' when the protocol provides similar semantics. However, this MUST be clearly defined in the specification that defines the interface protocol.

An Agent will respond with an ERROR, OK, or an OK WITH INDICATION that remaining data will be sent via a notify from the Agent to the Client Section 5.1.1.6.2 for CONFIG and CONF_BUNDLE requests. When returning an 'ok' of any kind, optional data may be present.

Two Agent notifications are supported:

Message	Type	Description
CONFIG_RESULT_NOTIFY	See Table 15	An asynchronous notification from Agent to Client based upon a previous CONFIG or CONF_BUNDLE request.
NOTIFY	See Table 16	An asynchronous notification from Agent to Client based upon a registered MONITOR.

Table 2: Agent to Client Messages (notifications)

5.1.1. CONFIG and CONF_BUNDLE Messages

CONFIG and CONF_BUNDLE specify the following information for each operation in addition to the header information:

SESSION_STATE: sets the expected state of the entities embedded in the operation body after successful completion of the operation. Values can be ’complete’, ’incomplete’ or ’outdated’. Any operation that is ’incomplete’ MAY NOT result in communication between the Agent and DPN. If the result is ’outdated’ any new operations on these entities or new references to these entities have unpredictable results.

OP_TYPE: specifies the type of operation. Valid values are ’create’ (0), ’update’ (1), ’query’ (2) or ’delete’ (3).

COMMAND_SET: If the feature is supported, specifies the Command Set (see Section 5.1.1.4).

BODY: A list of Clones, if supported, Vports and Contexts when the OP_TYPE is ’create’ or ’update’. Otherwise it is a list of Targets for ’query’ or ’deletion’. See Section 6.2.2 for details.

5.1.1.1. Agent Operation Processing

The Agent will process entities provided in an operation in the following order:

1. Clone Instructions, if the feature is supported

2. Vports

3. Contexts according to COMMAND_SET order processing

The following Order Processing occurs when COMMAND Sets are present

1. The Entity-specific COMMAND_SET is processed according to its bit order unless otherwise specified by the technology specific COMMAND_SET definition.

2. Operation specific COMMAND_SET is processed upon all applicable entities (even if they had Entity-specific COMMAND_SET values present) according to its bit order unless otherwise specified by the technology specific COMMAND_SET definition.

3. Operation OP_TYPE is processed for all entities.

When deleting objects only their name needs to be provided. However, attributes MAY be provided if the Client wishes to avoid requiring the Agent cache lookups.

When deleting an attribute, a leaf reference should be provided. This is a path to the attributes.

5.1.1.2. Policy RPC Support

This optional feature permits policy elements, (Policy-Group, Policy, Action and Descriptor), values to be in CONFIG or CONF_BUNDLE requests. It enables RPC based policy provisioning.

5.1.1.3. Cloning

Cloning is an optional feature that allows a Client to copy one structure to another in an operation. Cloning is always done first within the operation (see Operation Order of Execution for more detail). If a Client wants to build an object then Clone it, use CONF_BUNDLE with the first operation being the entities to be copied and a second operation with the Cloning instructions. A CLONE operation takes two arguments, the first is the name of the target to clone and the second is the name of the newly created entity. Individual attributes are not clonable; only Vports and Contexts can be cloned.

5.1.1.4. Command Bitsets

The COMMAND_SET is a technology specific bitset that allows for a single entity to be sent in an operation with requested sub-transactions to be completed. For example, a Context could have the Home Network Prefix absent but it is unclear if the Client would like the address to be assigned by the Agent or if this is an error.

Rather than creating a specific command for assigning the IP a bit position in a COMMAND_SET is reserved for Agent based IP assignment. Alternatively, an entity could be sent in an update operation that would be considered incomplete, e.g. missing some required data in for the entity, but has sufficient data to complete the instructions provided in the COMMAND_SET.

5.1.1.5. Reference Scope

The Reference Scope is an optional feature that provides the scope of references used in a configuration command, i.e. CONFIG or CONF_BUNDLE. These scopes are defined as

o none - all entities have no references to other entities. This implies only Contexts are present. Vports MUST have references to Policy-Groups.

o op - All references are contained in the operation body, i.e. only intra-operation references exist.

o bundle - All references exist in bundle (inter-operation/intra-bundle). NOTE - If this value is present in a CONFIG message it is equivalent to 'op'.

o storage - One or more references exist outside of the operation and bundle. A lookup to a cache / storage is required.

o unknown - the location of the references are unknown. This is treated as a 'storage' type.

If supported by the Agent, when cloning instructions are present, the scope MUST NOT be 'none'. When Vports are present the scope MUST be 'storage' or 'unknown'.

An agent that only accepts 'op' or 'bundle' reference scope messages is referred to as 'stateless' as it has no direct memory of references outside messages themselves. This permits low memory footprint Agents. Even when an Agent supports all message types an 'op' or 'bundle' scoped message can be processed quickly by the Agent as it does not require storage access.

5.1.1.6. Operation Response

5.1.1.6.1. Immediate Response

Results will be supplied per operation input. Each result contains the RESULT_STATUS and OP_ID that it corresponds to. RESULT_STATUS values are:

OK - Success

ERR - An Error has occurred

OK_NOTIFY_FOLLOWS - The Operation has been accepted by the Agent but further processing is required. A CONFIG_RESULT_NOTIFY will be sent once the processing has succeeded or failed.

Any result MAY contain nothing or entities created or partially fulfilled as part of the operation as specified in Table 14. For Clients that need attributes back quickly for call processing, the AGENT MUST respond back with an OK_NOTIFY_FOLLOWS and minimally the attributes assigned by the Agent in the response. These situations MUST be determined through the use of Command Sets (see Section 5.1.1.4).

If an error occurs the following information is returned.

ERROR_TYPE_ID (Unsigned 32) - The identifier of a specific error type

ERROR_INFORMATION - An OPTIONAL string of no more than 1024 characters.

5.1.1.6.2. Asynchronous Notification

A CONFIG_RESULT_NOTIFY occurs after the Agent has completed processing related to a CONFIG or CONF_BUNDLE request. It is an asynchronous communication from the Agent to the Client.

The values of the CONFIG_RESULT_NOTIFY are detailed in Table 15.

5.1.2. Monitors

When a monitor has a reporting configuration of SCHEDULED it is automatically de-registered after the NOTIFY occurs. An Agent or DPN may temporarily suspend monitoring if insufficient resources exist. In such a case the Agent MUST notify the Client.

All monitored data can be requested by the Client at any time using the PROBE message. Thus, reporting configuration is optional and when not present only PROBE messages may be used for monitoring. If a SCHEDULED or PERIODIC configuration is provided during registration with the time related value (time or period respectively) of 0 a NOTIFY is immediately sent and the monitor is immediately de-registered. This method should, when a MONITOR has not been installed, result in an immediate NOTIFY sufficient for the Client’s needs and lets the Agent realize the Client has no further need for the monitor to be registered. An Agent may reject a registration if it or the DPN has insufficient resources.

PROBE messages are also used by a Client to retrieve information about a previously installed monitor. The PROBE message SHOULD identify one or more monitors by means of including the associated monitor identifier. An Agent receiving a PROBE message sends the requested information in a single or multiple NOTIFY messages.

5.1.2.1. Operation Response

5.1.2.1.1. Immediate Response

Results will be supplied per operation input. Each result contains the RESULT_STATUS and OP_ID that it corresponds to. RESULT_STATUS values are:

OK - Success

ERR - An Error has occurred

Any OK result will contain no more information.

If an error occurs the following information is returned.

ERROR_TYPE_ID (Unsigned 32) - The identifier of a specific error type

ERROR_INFORMATION - An OPTIONAL string of no more than 1024 characters.

5.1.2.1.2. Asynchronous Notification

A NOTIFY can be sent as part of de-registraiton, a trigger based upon a Monitor Configuration or a PROBE. A NOTIFY is comprised of unique Notification Identifier from the Agent, the Monitor ID the notification applies to, the Trigger for the notification, a timestamp of when the notification’s associated event occurs and data that is specific to the monitored value’s type.

5.2. Protocol Operation

5.2.1. Simple RPC Operation

An FPC Client and Agent MUST identify themselves using the CLI_ID and AGT_ID respectively to ensure that for all transactions a recipient of an FPC message can unambiguously identify the sender of the FPC message. A Client MAY direct the Agent to enforce a rule in a particular DPN by including a DPN_ID value in a Context. Otherwise the Agent selects a suitable DPN to enforce a Context and notifies the Client about the selected DPN using the DPN_ID.

All messages sent from a Client to an Agent MUST be acknowledged by the Agent. The response must include all entities as well as status information, which indicates the result of processing the message, using the RESPONSE_BODY property. In case the processing of the message results in a failure, the Agent sets the ERROR_TYPE_ID and ERROR_INFORMATION accordingly and MAY clear the Context or Vport, which caused the failure, in the response.

If based upon Agent configuration or the processing of the request possibly taking a significant amount of time the Agent MAY respond with an OK_NOTIFY_FOLLOWS with an optional RESPONSE_BODY containing the partially completed entities. When an OK_NOTIFY_FOLLOWS is sent, the Agent will, upon completion or failure of the operation, respond with an asynchronous CONFIG_RESULT_NOTIFY to the Client.

A Client MAY add a property to a Context without providing all required details of the attribute’s value. In such case the Agent SHOULD determine the missing details and provide the completed property description back to the Client. If the processing will take too long or based upon Agent configuration, the Agent MAY respond with an OK_NOTIFY_FOLLOWS with a RESPONSE_BODY containing the partially completed entities.

In case the Agent cannot determine the missing value of an attribute’s value per the Client’s request, it leaves the attribute’s value cleared in the RESPONSE_BODY and sets the RESULT to Error, ERROR_TYPE_ID and ERROR_INFORMATION. As example, the Control-Plane needs to setup a tunnel configuration in the Data-Plane but has to rely on the Agent to determine the tunnel endpoint which is associated with the DPN that supports the Context. The Client adds the tunnel property attribute to the FPC message and clears the value of the attribute (e.g. IP address of the local tunnel endpoint). The Agent determines the tunnel endpoint and includes the completed tunnel property in its response to the Client.

Figure 17 illustrates an exemplary session life-cycle based on Proxy Mobile IPv6 registration via MAG Control-Plane function 1 (MAG-C1) and handover to MAG Control-Plane function 2 (MAG-C2). Edge DPN1 represents the Proxy CoA after attachment, whereas Edge DPN2 serves as Proxy CoA after handover. As exemplary architecture, the FPC Agent and the network control function are assumed to be co-located with the Anchor-DPN, e.g. a Router.

```
                                                 +-------Router--------+
                           +-----------+         |+-------+ +---------+|
   +------+ +------+     +-----+ FPC   |          | FPC   | |  Anchor |
   |MAG-C1| |MAG-C2|     |LMA-C| Client|          | Agent | |   DPN   |
   +------+ +------+     +-----+-------+          +-------+ +---------+
   [MN attach]  |             |                          |           |
      |-------------PBU----->|                          |           |
      |         |            |---(1)--CONFIG(CREATE)--->|           |
      |         |            |   [ CONTEXT_ID,          |--tun1 up->|
      |         |            |   DOWNLINK(QOS/TUN),     |           |
      |         |            |   UPLINK(QOS/TUN),       |--tc qos-->|
      |         |            |     IP_PREFIX(HNP) ]     |           |
      |         |            |<---(2)- OK --------------|-route add>|
      |         |            |                          |           |
      |<------------PBA------|                          |           |
      |         |            |                          |           |
      | +----+  |            |                          |           |
      | |Edge|  |            |                          |           |
      | |DPN1|  |            |                          |           |
      | +----+  |            |                          |           |
      |   |                                                         |
      |   |-=======================================================-|
      |                      |                          |           |
      |   [MN handover]      |                          |           |
      |         |---PBU ---->|                          |           |
      |         |            |--(3)- CONFIG(MODIFY)---->|           |
      |         |<--PBA------|    [ CONTEXT_ID          |-tun1 mod->|
      |         |            |      DOWNLINK(TUN),      |           |
      |         |  +----+    |      UPLINK(TUN) ]       |           |
      |         |  |Edge|    |<---(4)- OK --------------|           |
      |         |  |DPN2|    |                          |           |
      |         |  +----+    |                          |           |
      |         |    |       |                          |           |
      |         |    |-============================================-|
      |         |            |                          |           |
```

Figure 17: Exemplary Message Sequence (focus on FPC reference point)

After reception of the Proxy Binding Update (PBU) at the LMA Control-Plane function (LMA-C), the LMA-C selects a suitable DPN, which serves as Data-Plane anchor to the mobile node’s (MN) traffic. The LMA-C adds a new logical Context to the DPN to treat the MN’s traffic (1) and includes a Context Identifier (CONTEXT_ID) to the CONFIG command. The LMA-C identifies the selected Anchor DPN by including the associated DPN identifier.

The LMA-C adds properties during the creation of the new Context. One property is added to specify the forwarding tunnel type and endpoints (Anchor DPN, Edge DPN1) in each direction (as required). Another property is added to specify the QoS differentiation, which the MN's traffic should experience. At reception of the Context, the FPC Agent utilizes local configuration commands to create the tunnel (tun1) as well as the traffic control (tc) to enable QoS differentiation. After configuration has been completed, the Agent applies a new route to forward all traffic destined to the MN's HNP specified as a property in the Context to the configured tunnel interface (tun1).

During handover, the LMA-C receives an updating PBU from the handover target MAG-C2. The PBU refers to a new Data-Plane node (Edge DPN2) to represent the new tunnel endpoints in the downlink and uplink, as required. The LMA-C sends a CONFIG message (3) to the Agent to modify the existing tunnel property of the existing Context and to update the tunnel endpoint from Edge DPN1 to Edge DPN2. Upon reception of the CONFIG message, the Agent applies updated tunnel property to the local configuration and responds to the Client (4).

```
                                                 +-------Router--------+
                         +-----------+           |+-------+ +---------+|
+------+ +------+     +-----+ FPC    |            | FPC   | |  Anchor |
|MAG-C1| |MAG-C2|     |LMA-C| Client|            | Agent | |   DPN   |
+------+ +------+     +-----+-------+            +-------+ +---------+
[MN attach]  |             |                           |            |
   |-------------PBU----->|                           |            |
   |          |            |---(1)--CONFIG(MODIFY)--->|            |
   |<------------PBA------|   [ CONTEXT_ID,            |--tun1   ->|
   |          |            |   DOWNLINK(TUN delete),   |    down    |
   |          |            |   UPLINK(TUN delete) ]    |            |
   |          |            |                           |            |
   |          |            |<-(2)- OK ----------------|            |
   |          |            |                           |            |
   |          |  [ MinDelayBeforeBCEDelete expires ]  |            |
   |          |            |                           |            |
   |          |            |---(3)--CONFIG(DELETE)--->|-- tun1 -->|
   |          |            |                           |  delete    |
   |          |            |<-(4)- OK ----------------|            |
   |          |            |                           |-- route ->|
   |          |            |                           |   remove   |
   |          |            |                           |            |
```

Figure 18: Exemplary Message Sequence (focus on FPC reference point)

When a teardown of the session occurs, MAG-C1 will send a PBU with a lifetime value of zero. The LMA-C sends a CONFIG message (1) to the

Agent to modify the existing tunnel property of the existing Context to delete the tunnel information.) Upon reception of the CONFIG message, the Agent removes the tunnel configuration and responds to the Client (2). Per [RFC5213], the PBA is sent back immediately after the PBA is received.

If no valid PBA is received after the expiration of the MinDelayBeforeBCEDelete timer (see [RFC5213]), the LMA-C will send a CONFIG (3) message with a deletion request for the Context. Upon reception of the message, the Agent deletes the tunnel and route on the DPN and responds to the Client (4).

When a multi-DPN Agent is used the DPN list permits several DPNs to be provisioned in a single message.

```
                      +-----------+           +-------+ +---------+
+------+ +------+     +-----+ FPC   |           | FPC   | |  Anchor |
|MAG-C1| |MAG-C2|     |LMA-C| Client|           | Agent | |   DPN1  |
+------+ +------+     +-----+-------+           +-------+ +---------+
[MN attach]  |            |                          |            |
   |-------------PBU----->|                          |            |
   |         |            |---(1)--CONFIG(CREATE)--->|            |
   |         |            |   [ CONTEXT_ID, DPNS [   |--tun1 up->|
   |         |            |[DPN1,DOWNLINK(QOS/TUN)], |            |
   |         |            | [DPN1,UPLINK(QOS/TUN)],  |--tc qos-->|
   |         |            |[DPN2,DOWNLINK(QOS/TUN)], |            |
   |         |            | [DPN2,UPLINK(QOS/TUN)],  |            |
   |         |            |     IP_PREFIX(HNP) ]     |            |
   |         |            |<-(2)- OK_NOTIFY_FOLLOWS -|-route add>|
   |         |            |                          |            |
   |<------------PBA------|                          |            |
   |         |            |                          |            |
   | +----+               |                          |            |
   | |Edge|               |                          |            |
   | |DPN2|               |                          |            |
   | +----+               |                          |            |
   |   |<--------------------- tun1 up -------------|            |
   |   |<--------------------- tc qos --------------|            |
   |   |<--------------------- route add -----------|            |
   |   |                  |                          |            |
   |   |                  |<(3) CONFIG_RESULT_NOTIFY |            |
   |   |                  |   [ Response Data ]      |            |
   |   |                  |                          |            |
```

Figure 19: Exemplary Message Sequence for Multi-DPN Agent

Figure 19 shows how the first 2 messages in Figure 17 are supported when a multi-DPN Agent communicates with both Anchor DPN1 and Edge DPN2. In such a case, the FPC Client sends the downlink and uplink for both DPNs in the "DPNS" list of the same Context. Message 1 shows the DPNS list with all entries. Each entry identifies the DPN and direction (one of ’uplink’, ’downlink’ or ’both’). Generally, the ’both’ direction is not used for normal mobility session processing. It is commonly used for the instantiation of Policies on a specific DPN (see Section 5.2.4).

The Agent responds with an OK_NOTIFY_FOLLOWS while it simultaneoulsy provisions both DPNs. Upon successful completion, the Agent responds to the Client with a CONFIG_RESULT_NOTIFY indicating the operation status.

5.2.2. Policy And Mobility on the Agent

A Client may build Policy and Topology using any mechanism on the Agent. Such entities are not always required to be constructed in realtime and, therefore, there are no specific messages defined for them in this specification.

The Client may add, modify or delete many Vports and Contexts in a single FPC message. This includes linking Contexts to Actions and Descriptors, i.e. a Rule. As example, a Rule which performs re-writing of an arriving packet’s destination IP address from IP_A to IP_B matching an associated Descriptor, can be enforced in the Data-Plane via an Agent to implicitly consider matching arriving packet’s source IP address against IP_B and re- write the source IP address to IP_A.

Figure 20 illustrates the generic policy configuration model as used between a FPC Client and a FPC Agent.

```
           Descriptor_1 -+          +- Action_1
                         |          |
           Descriptor_2 -+--<Rule>--+- Action_2
                           +------+
                           /Order#/-------------+
                           +------+             |
                                                |
           Descriptor_3 -+          +- Action_3 +-<PolicyID>
                         |          |           |  ^
           Descriptor_4 -+--<Rule>--+- Action_4 |  |
                           +------+             | <PolicyGroupID>
                           /Order#/-------------+  ^
                           +------+                |
                                                  <VportID>

           +-------------------+     +---------------------+
           | Bind 1..M traffic |     |  Bind 1..N traffic  |
           |  Descriptors to   | --> |  treatment actions  |
           |     a Policy,     |     |      to a Policy,   |
           | Policy-Group and  |     |   Policy-Group and  |
           |       Vport       |     |       Vport         |
           +-------------------+     +---------------------+

          |                                                 |
          +-------------- Data-Plane Rule ------------------+
```

Figure 20: Structure of Policies and Vports

As depicted in Figure 20, the Vport represents the anchor of Rules through the Policy-group, Policy, Rule hierarchy configured by any mechanism including RPC or N. A Client and Agent use the identifier of the associated Policy to directly access the Rule and perform modifications of traffic Descriptors or Action references. A Client and Agent use the identifiers to access the Descriptors or Actions to perform modifications. From the viewpoint of packet processing, arriving packets are matched against traffic Descriptors and processed according to the treatment Actions specified in the list of properties associated with the Vport.

A Client complements a rule’s Descriptors with a Rule’s Order (priority) value to allow unambiguous traffic matching on the Data-Plane.

Figure 21 illustrates the generic context configuration model as used between a FPC Client and a FPC Agent.

```
             TrafficSelector_1
                      |
             profile-parameters
                      |
             mobility-profile-- dl ------+
                                ^        |
                                |      qos-profile
                           <ContextID1>       |
                                ^        per-mn-agg-max-dl_2
                                |
                           <ContextID2>

             +-------------------+     +---------------------+
             | Bind 1..M traffic |     |  Bind 1..N traffic  |
             |    selectors to   | --> |  treatment / qos    |
             |     a Context     |     |  actions to a       |
             |                   |     |       Context       |
             +-------------------+     +---------------------+

            |                                                 |
            +-------------- Data-Plane Rule ------------------+
```

Figure 21: Structure of Contexts

As depicted in Figure 21, the Context represents a mobility session hierarchy. A Client and Agent directly assigns values such as downlink traffic descriptors, QoS information, etc. A Client and Agent use the context identifiers to access the descriptors, qos information, etc. to perform modifications. From the viewpoint of packet processing, arriving packets are matched against traffic Descriptors and processed according to the qos or other mobility profile related Actions specified in the Context's properties. If present, the final action is to use a Context's tunnel information to encapsulate and forward the packet.

A second Context also references context1 in the figure. Based upon the technology a property in a parent context MAY be inherited by its descendants. This permits concise over the wire representation. When a Client deletes a parent Context all children are also deleted.

5.2.3. Optimization for Current and Subsequent Messages

5.2.3.1. Bulk Data in a Single Operation

A single operation MAY contain multiple entities. This permits bundling of requests into a single operation. In the example below two PMIP sessions are created via two PBU messages and sent to the Agent in a single CONFIG message (1). Upon recieveing the message, the Agent responds back with an OK_NOTIFY_FOLLOWS (2), completes work on the DPN to activate the associated sessions then responds to the Client with a CONFIG_RESULT_NOTIFY (3).

```
                                                 +-------Router--------+
                           +-----------+         |+-------+ +---------+|
   +------+ +------+     +-----+ FPC   |          | FPC   | |  Anchor |
   |MAG-C1| |MAG-C2|     |LMA-C| Client|          | Agent | |   DPN   |
   +------+ +------+     +-----+-------+          +-------+ +---------+
   [MN1 attach] |            |                          |           |
      |-------------PBU----->|                          |           |
      |  [MN2 attach]        |                          |           |
      |         |---PBU----->|                          |           |
      |         |            |                          |           |
      |         |            |---(1)--CONFIG(CREATE)--->|           |
      |<------------PBA------|   [ CONTEXT_ID 1,        |--tun1 up->|
      |         |            |   DOWNLINK(QOS/TUN),     |           |
      |         |<--PBA------|   UPLINK(QOS/TUN),       |--tc1 qos->|
      |         |            |     IP_PREFIX(HNP) ]     |           |
      |         |            |   [ CONTEXT_ID 2,        |-route1    |
      |         |            |   DOWNLINK(QOS/TUN),     |   add>    |
      |         |            |   UPLINK(QOS/TUN),       |           |
      |         |            |     IP_PREFIX(HNP) ]     |--tun2 up->|
      |         |            |<-(2)- OK_NOTIFY_FOLLOWS--|           |
      |         |            |                          |--tc2 qos->|
      |<------------PBA------|                          |           |
      |         |            |                          |-route2    |
      |         |            |<(3) CONFIG_RESULT_NOTIFY |   add>    |
      |         |            |   [ Response Data ]      |           |
      |         |            |                          |           |
      |         |            |                          |           |
```

Figure 22: Exemplary Bulk Entity with Asynchronous Notification Sequence (focus on FPC reference point)

5.2.3.2. Configuration Bundles

Bundles provide transaction boundaries around work in a single message. Operations in a bundle MUST be successfully executed in the order specified. This allows references created in one operation to be used in a subsequent operation in the bundle.

The example bundle shows in Operation 1 (OP 1) the creation of a Context 1 which is then referenced in Operation 2 (OP 2) by CONTEXT_ID 2. If OP 1 fails then OP 2 will not be executed. The advantage of the CONF_BUNDLE is preservation of dependency orders in a single message as opposed to sending multiple CONFIG messages and awaiting results from the Agent.

When a CONF_BUNDLE fails, any entities provisioned in the CURRENT operation are removed, however, any successful operations completed prior to the current operation are preserved in order to reduce system load.

```
                             +-------Router--------+
      +-----------+          |+-------+ +---------+|
      |   FPC     |           | FPC   | |  Anchor |
      |  Client   |           | Agent | |   DPN   |
      +-----------+           +-------+ +---------+
           |                           |            |
           |--CONF_BUNDLE(CREATE)----->|            |
           |   [ OP 1, [VPORT X ]      |            |
           |   [ CONTEXT_ID 1,         |            |
           |   DOWNLINK(QOS/TUN),      |            |
           |   UPLINK(QOS/TUN),        |            |
           |     IP_PREFIX(HNP) ]      |            |
           |   [ OP 2,                 |            |
           |    [ CONTEXT_ID 2,        |            |
           |   PARENT_CONTEXT_ID 1,    |            |
           |   UPLINK(QOS/TUN),        |            |
           |   DOWNLINK(QOS/TUN) ] ]   |            |
           |                           |            |
```

Figure 23: Exemplary Bundle Message (focus on FPC reference point)

5.2.3.3. Cloning Feature (Optional)

Cloning provides a high speed copy/paste mechanism. The example below shows a single Context that will be copied two times. A subsequent update will then override copied values. To avoid the accidental activation of the Contexts on the DPN, the CONFIG (1) message with the cloning instruction has a SESSION_STATE with a value of ’incomplete’ and OP_TYPE of ’CREATE’. A second CONFIG (2) is sent with the SESSION_STATE of ’complete’ and OP_TYPE of ’UPDATE’. The second message includes any differences between the original (copied) Context and its Clones.

```
                                      +-------Router--------+
              +-----------+           |+-------+ +---------+|
              |   FPC     |            | FPC   | |  Anchor |
              |  Client   |            | Agent | |   DPN   |
              +-----------+            +-------+ +---------+
                    |                          |           |
                    |--CONF_BUNDLE(CREATE)---->|           |
                    |   [ OP 1,                |           |
                    |    [ SESSION_STATE       |           |
                    |       (incomplete) ],    |           |
                    | [CLONE SRC=2, TARGET=3], |           |
                    | [CLONE SRC=2, TARGET=4], |           |
                    |    [ CONTEXT_ID 2,       |           |
                    |   PARENT_CONTEXT_ID 1,   |           |
                    |   UPLINK(QOS/TUN),       |           |
                    |   DOWNLINK(QOS/TUN),     |           |
                    |   IP_PREFIX(HNP)    ] ]  |           |
                    |<----- OK ----------------|           |
                    |                          |           |
                    |--CONF_BUNDLE(UPDATE)--->|           |
                    |    [ CONTEXT_ID 3,       |           |
                    | PARENT_CONTEXT_ID(empty),|           |
                    |   UPLINK(QOS/TUN),       |           |
                    |   DOWNLINK(QOS/TUN) ],   |           |
                    |    [ CONTEXT_ID 4,       |           |
                    | PARENT_CONTEXT_ID(empty),|           |
                    |   UPLINK(QOS/TUN),       |           |
                    |   DOWNLINK(QOS/TUN) ] ]  |           |
                    |<----- OK ----------------|           |
                    |                          |           |
```

Figure 24: Exemplary Bundle Message (focus on FPC reference point)

Cloning has the added advantage of reducing the over the wire data size required to create multiple entities. This can improve performance if serialization / deserialization of multiple entities incurs some form of performance penalty.

5.2.3.4. Command Bitsets (Optional)

Command Sets permit the ability to provide a single, unified data structure, e.g. CONTEXT, and specify which activities are expected to be performed on the DPN. This has some advantages

o Rather than sending N messages with a single operation performed on the DPN a single message can be used with a Command Set that specifies the N DPN operations to be executed.

- Errors become more obvious. For example, if the HNP is NOT provided but the Client did not specify that the HNP should be assigned by the Agent this error is easily detected. Without the Command Set the default behavior of the Agent would be to assign the HNP and then respond back to the Client where the error would be detected and subsequent messaging would be required to remedy the error. Such situations can increase the time to error detection and overall system load without the Command Set present.

- Unambiguous provisioning specification. The Agent is exactly in sync with the expectations of the Client as opposed to guessing what DPN work could be done based upon data present at the Agent. This greatly increases the speed by which the Agent can complete work.

- Permits different technologies with different instructions to be sent in the same message.

As Command Bitsets are technology specific, e.g. PMIP or 3GPP Mobility, the type of work varies on the DPN and the amount of data present in a Context or Port will vary. Using the technology specific instructions allows the Client to serve multiple technologies and MAY result in a more stateless Client as the instructions are transferred the Agent which will match the desired, technology specific instructions with the capabilities and over the wire protocol of the DPN more efficiently.

5.2.3.5. Reference Scope(Optional)

Although entities MAY refer to any other entity of an appropriate type, e.g. Contexts can refer to Vports or Contexts, the Reference Scope gives the Agent an idea of where those references reside. They may be in the same operation, an operation in the same CONF_BUNDLE message or in storage. There may also be no references. This permits the Agent to understand when it can stop searching for reference it cannot find. For example, if a CONF_BUNDLE message uses a Reference Scope of type ’op’ then it merely needs to keep an operation level cache and consume no memory or resources searching across the many operations in the CONF_BUNDLE message or the data store.

Agents can also be stateless by only supporting the ’none’, ’op’ and ’bundle’ reference scopes. This does not imply they lack storage but merely the search space they use when looking up references for an entity. The figure below shows the caching hierarchy provided by the Reference Scope

Caches are temporarily created at each level and as the scope includes more caches the amount of entities that are searched increases. Figure 25 shows an example containment hierarchy provided for all caches.

```
                             +---------------+
                             | Global Cache  |
                             |  (storage)    |
                             +------+--------+
                                    |
                                    +----------------------+
                                    |                      |
                             +------+--------+      +------+--------+
                             | Bundle Cache  |      | Bundle Cache  |
                             |   (bundle)    | .... |   (bundle)    |
                             +------+--------+      +------+--------+
                                    |
               +--------------------+--------------------+
               |                    |                    |
      +--------+---------+ +--------+---------+ +--------+---------+
      | Operation Cache  | | Operation Cache  | | Operation Cache  |
      |       (op)       | |       (op)       | |       (op)       |
      +------------------+ +------------------+ +------------------+

                                (no cache)
```

Figure 25: Exemplary Hierarchical Cache

5.2.4. Pre-provisioning

Although Contexts are used for Session based lifecycle elements, Vports may exist outside of a specific lifecycle and represent more general policies that may affect multiple Contexts (sessions). The use of pre-provisioning of Vports permits policy and administrative use cases to be executed. For example, creating tunnels to forward traffic to a trouble management platform and dropping packets to a defective web server can be accomplished via provisioning of Vports.

The figure below shows a CONFIG (1) message used to install a Policy-group, policy-group1, using a Context set aside for pre-provisioning on a DPN.

```
                                 +-------Router--------+
              +-----------+      |+-------+ +---------+|
              |   FPC     |       | FPC   | |  Anchor |
              |  Client   |       | Agent | |   DPN   |
              +-----------+       +-------+ +---------+
                    |                       |            |
                    |------CONFIG(CREATE)----->|            |
                    |  [ VPORT_ID port1,       |            |
                    |     [ policy-group1 ] ]  |            |
                    |  [ CONTEXT_ID preprov,   |            |
                    |     DPN_ID X,            |            |
                    |     [ port1 ] ]          |            |
                    |                       |            |
```

Figure 26: Exemplary Config Message for policy pre-provisioning

5.2.4.1. Basename Registry Feature (Optional)

The Optional BaseName Registry support feature is provided to permit Clients and tenants with common scopes, referred to in this specification as BaseNames, to track the state of provisioned policy information on an Agent. The registry records the BaseName and Checkpoint set by a Client. If a new Client attaches to the Agent it can query the Registry to determine the amount of work that must be executed to configure the Agent to a BaseName / checkpoint revision. A State value is also provided in the registry to help Clients coordinate work on common BaseNames.

6. Protocol Message Details

6.1. Data Structures And Type Assignment

6.1.1. Policy Structures

Structure	Field	Type
ACTION	ACTION_ID	FPC-Identity (Section 4.4)
ACTION	TYPE	[32, unsigned integer]
ACTION	VALUE	Type specific
DESCRIPTOR	DESCRIPTOR_ID	FPC-Identity (Section 4.4)
DESCRIPTOR	TYPE	[32, unsigned integer]
DESCRIPTOR	VALUE	Type specific
POLICY	POLICY_ID	FPC-Identity (Section 4.4)
POLICY	RULES	*[RULE] (See Table 4)
POLICY-GROUP	POLICY_GROUP_ID	FPC-Identity (Section 4.4)
POLICY-GROUP	POLICIES	*[POLICY_ID]

Table 3: Action Fields

Policies contain a list of Rules by their order value. Each Rule contains Descriptors with optional directionality and Actions with order values that specifies action execution ordering if the Rule has multiple actions.

Rules consist of the following fields.

Field	Type	Sub-Fields
ORDER	[16, INTEGER]	
RULE_DESCRIPTORS	*[DESCRIPTOR_ID DIRECTION]	DIRECTION [2, unsigned bits] is an ENUMERATION (uplink, downlink or both).
RULE_ACTIONS	*[ACTION_ID ACTION-ORDER]	ACTION-ORDER [8, unsigned integer] specifies action execution order.

Table 4: Rule Fields

6.1.2. Mobility Structures

Field	Type
VPORT_ID	FPC-Identity (Section 4.4)
POLICIES	*[POLICY_GROUP_ID]

Table 5: Vport Fields

Field	Type
CONTEXT_ID	FPC-Identity (Section 4.4)
VPORTS	*[VPORT_ID]
DPN_GROUP_ID	FPC-Identity (Section 4.4)
DELEGATED IP PREFIXES	*[IP_PREFIX]
PARENT_CONTEXT_ID	FPC-Identity (Section 4.4)
UPLINK [NOTE 1]	MOB_FIELDS
DOWNLINK [NOTE 1]	MOB_FIELDS
DPNS [NOTE 2]	*[DPN_ID DPN_DIRECTION MOB_FIELDS]
MOB_FIELDS	All parameters from Table 7

Table 6: Context Fields

NOTE 1 - These fields are present when the Agent supports only a single DPN.

NOTE 2 - This field is present when the Agent supports multiple DPNs.

Field	Type	Detail
TUN_LOCAL_ADDRESS	IP Address	[NOTE 1]
TUN_REMOTE_ADDRESS	IP Address	[NOTE 1]

TUN_MTU	[32, unsigned integer]	
TUN_PAYLOAD_TYPE	[2, bits]	Enumeration: payload_ipv4(0), payload_ipv6(1) or payload_dual (2).
TUN_TYPE	[8, unsigned integer]	Enumeration: IP-in-IP(0), UDP(1), GRE(2) and GTP(3).
TUN_IF	[16, unsigned integer]	Input interface index.
MOBILITY_SPECIFIC_TUN_PARAMS	[IETF_PMIP_MOB_PROFILE | 3GPP_MOB_PROFILE]	[NOTE 1]
NEXTHOP	[IP Address | MAC Address | SPI | MPLS Label | SID | Interface Index] (See Table 19).	[NOTE 1]
QOS_PROFILE_PARAMS	[3GPP_QOS | PMIP_QOS]	[NOTE 1]
DPN_SPECIFIC_PARAMS	[TUN_IF or Varies]	Specifies optional node specific parameters in need such as if-index, tunnel-if-number that must be unique in the DPN.
VENDOR_SPECIFIC_PARAM	*[Varies]	[NOTE 1]

NOTE 1 - These parameters are extensible. The Types may be extended for Field value by future specifications or in the case of Vendor Specific Attributes by enterprises.

Table 7: Context Downlink/Uplink Field Definitions

6.1.3. Topology Structures

Field	Type
DPN_ID	FPC-Identity. See Section 4.4
DPN_NAME	[1024, OCTET STRING]
DPN_GROUPS	* [FPC-Identity] See Section 4.4
NODE_REFERENCE	[1024, OCTET STRING]

Table 8: DPN Fields

Field	Type
DOMAIN_ID	[1024, OCTET STRING]
DOMAIN_NAME	[1024, OCTET STRING]
DOMAIN_TYPE	[1024, OCTET STRING]
DOMAIN_REFERENCE	[1024, OCTET STRING]

Table 9: Domain Fields

Field	Type
DPN_GROUP_ID	FPC-Identity. See Section 4.4
DATA_PLANE_ROLE	[4, ENUMERATION (data-plane, such as access-dpn, L2/L3 anchor-dpn.)]
ACCESS_TYPE	[4, ENUMERATION ()ethernet(802.3/11), 3gpp cellular(S1,RAB)]
MOBILITY_PROFILE	[4, ENUMERATION (ietf-pmip, 3gpp, or new profile)]
PEER_DPN_GROUPS	* [DPN_GROUP_ID MOBILITY_PROFILE REMOTE_ENDPOINT_ADDRESS LOCAL_ENDPOINT_ADDRESS TUN_MTU DATA_PLANE_ROLE]

Table 10: DPN Groups Fields

6.1.4. Monitors

Field	Type	Description
MONITOR	MONITOR_ID TARGET [REPORT_CONFIG]	
MONITOR_ID	FPC-Identity. See Section 4.4	
EVENT_TYPE_ID	[8, Event Type ID]	Event Type (unsigned integer).
TARGET	OCTET STRING (See Section 4.3.3)	
REPORT_CONFIG	[8, REPORT-TYPE] [TYPE_SPECIFIC_INFO]	
PERIODIC_CONFIG	[32, period]	report interval (ms).
THRESHOLD_CONFIG	[32, low] [32, hi]	thresholds (at least one value must be present)
SCHEDULED_CONFIG	[32, time]	
EVENTS_CONFIG	*[EVENT_TYPE_ID]	

Table 11: Monitor Structures and Attributes

TRIGGERS include but are not limited to the following values:

o Events specified in the Event List of an EVENTS CONFIG

o LOW_THRESHOLD_CROSSED

o HIGH_THRESHOLD_CROSSED

o PERIODIC_REPORT

o SCHEDULED_REPORT

o PROBED

o DEREG_FINAL_VALUE

6.2. Message Attributes

6.2.1. Header

Each operation contains a header with the following fields:

Field	Type	Messages
CLIENT_ID	FPC-Identity (Section 4.4)	All
DELAY	[32, unsigned integer]	All
OP_ID	[64, unsigned integer]	All
ADMIN_STATE	[8, admin state]	CONFIG, CONF_BUNDLE and REG_MONITOR
OP_TYPE	[8, op type]	CONFIG and CONF_BUNDLE

Table 12: Message Header Fields

6.2.2. CONFIG and CONF_BUNDLE Attributes and Notifications

Field	Type	Operation Types Create(C), Update(U), Query(Q) and Delete(D)
SESSION_STATE	[8, session state]	C,U
COMMAND_SET	FPC Command Bitset. See Section 5.1.1.4.	C,U [NOTE 1]
CLONES	*[FPC-Identity FPC-Identity] (Section 4.4)	C,U [NOTE 1]
VPORTS	*[VPORT]	C,U
CONTEXTS	*[CONTEXT [COMMAND_SET [NOTE 1]]]	C,U
TARGETS	FPC-Identity (Section 4.4) *[DPN_ID]	Q,D
POLICY_GROUPS	*[POLICY-GROUP]	C,U [NOTE 1]
POLICIES	*[POLICY]	C,U [NOTE 1]
DESCRIPTORS	*[DESCRIPTOR]	C,U [NOTE 1]
ACTIONS	*[ACTION]	C,U [NOTE 1]

NOTE 1 - Only present if the corresponding feature is supported by the Agent.

Table 13: CONFIG and CONF_BUNDLE OP_BODY Fields

Field	Type	Operation Types Create(C), Update(U), Query(Q) and Delete(D)
VPORTS	*[VPORT]	C,U [NOTE 2]
CONTEXTS	*[CONTEXT [COMMAND_SET [NOTE 1]]]	C,U [NOTE 2]
TARGETS	*[FPC-Identity (Section 4.4) *[DPN_ID]]	Q,D [NOTE 2]
ERROR_TYPE_ID	[32, unsigned integer]	All [NOTE 3]
ERROR_INFORMATION	[1024, octet string]	All [NOTE 3]

Table 14: Immediate Response RESPONSE_BODY Fields

Notes:

NOTE 1 - Only present if the corresponding feature is supported by the Agent.

NOTE 2 - Present in OK and OK_NOTIFY_FOLLOWS for both CONFIG and CONF_BUNDLE. MAY also be present in an CONF_BUNDLE Error response (ERR) if one of the operations completed successfully.

NOTE 3 - Present only for Error (ERR) responses.

Field	Type	Description
AGENT_ID	FPC-Identity (Section 4.4)	
NOTIFICATION_ID	[32, unsigned integer]	A Notification Identifier used to determine notification order.
TIMESTAMP	[32, unsigned integer]	The time that the notification occurred.
DATA	*[OP_ID RESPONSE_BODY (Table 14)]	

Table 15: CONFIG_RESULT_NOTIFY Asynchronous Notification Fields

6.2.3. Monitors

Field	Type	Description
NOTIFICATION_ID	[32, unsiged integer]	
TRIGGER	[32, unsigned integer]	
NOTIFY	NOTIFICATION_ID MONITOR_ID TRIGGER [32, timestamp] [NOTIFICATION_DATA]	Timestamp notes when the event occurred. Notification Data is TRIGGER and Monitor type specific.

Table 16: Monitor Notifications

7. Derived and Subtyped Attributes

This section notes derived attributes.

Field	Type Value	Type	Description
TO_PREFIX	0	[IP Address] [Prefix Len]	Aggregated or per-host destination IP address/prefix descriptor.
FROM_PREFIX	1	[IP Address] [Prefix Len]	Aggregated or per-host source IP address/prefix descriptor.
TRAFFIC_SELECTOR	2	Format per specification [RFC6088].	Traffic Selector.

Table 17: Descriptor Subtypes

Field	Type Value	Type	Description
DROP	0	Empty	Drop the associated packets.
REWRITE	1	[in_src_ip] [out_src_ip] [in_dst_ip] [out_dst_ip] [in_src_port] [out_src_port] [in_dst_port] [out_dst_port]	Rewrite IP Address (NAT) or IP Address / Port (NAPT).
COPY_FORWARD	2	FPC-Identity. See Section 4.4.	Copy all packets and forward them to the provided identity. The value of the identity MUST be a port or context.

Table 18: Action Subtypes

Field	Type Value	Type	Description
IP_ADDR	0	IP Address	An IP Address.
MAC_ADDR	1	MAC Address	A MAC Address.
SERVICE_PATH_ID	2	[24, unsigned integer]	Service Path Identifier (SPI)
MPLS_LABEL	3	[20, unsigned integer]	MPLS Label
NSH	4	[SERVICE_PATH_ID] [8, unsigned integer]	Included NSH which is a SPI and Service Index (8 bits).
INTERFACE_INDEX	5	[16, unsigned integer]	Interface Index (an unsigned integer).
SEGMENT_ID	5	[128, unsigned integer]	Segement Identifier.

Table 19: Next Hop Subtypes

Field	Type Value	Type	Description
QOS	0	[qos index type] [index] [DSCP]	Refers to a single index and DSCP to write to the packet.
GBR	1	[32, unsigned integer]	Guaranteed bit rate.
MBR	2	[32, unsigned integer]	Maximum bit rate.
PMIP_QOS	3	Varies by Type	A non-traffic selector PMIP QoS Attribute per [RFC7222]

Table 20: QoS Subtypes

Field	Type Value	Type	Description
IPIP_TUN	0		IP in IP Configuration
UDP_TUN	1	[src_port] [dst_port]	UDP Tunnel - source and/or destination port
GRE_TUN	2	[32, GRE Key]	GRE Tunnel.

Table 21: Tunnel Subtypes

The following COMMAND_SET values are supported for IETF_PMIP.

o assign-ip - Assign the IP Address for the mobile session.

o assign-dpn - Assign the Dataplane Node.

o session - Assign values for the Session Level.

o uplink - Command applies to uplink.

o downlink - Command applies to downlink.

7.1. 3GPP Specific Extenstions

3GPP support is optional and detailed in this section. The following acronyms are used:

APN-AMBR:	Access Point Name Aggregate Maximum Bit Rate
ARP:	Allocation of Retention Priority
EBI:	EPS Bearer Identity
GBR:	Guaranteed Bit Rate
GTP:	GPRS (General Packet Radio Service) Tunneling Protocol
IMSI:	International Mobile Subscriber Identity
MBR:	Maximum Bit Rate
QCI:	QoS Class Identifier
TEID:	Tunnel Endpoint Identifier.

TFT: Traffic Flow Template (TFT)

UE-AMBR: User Equipment Aggregate Maximum Bit Rate

NOTE: GTP Sequence Number (SEQ_NUMBER) is used in failover and handover.

Field	Type Value	Namespace / Entity Extended	Type
GTPV1	3	Tunnel Subtypes namespace.	LOCAL_TEID REMOTE_TEID SEQ_NUMBER
GTPV2	4	Tunnel Subtypes namespace.	LOCAL_TEID REMOTE_TEID SEQ_NUMBER
LOCAL_TEID	N/A	N/A	[32, unisgned integer]
REMOTE_TEID	N/A	N/A	[32, unisgned integer]
SEQ_NUMBER	N/A	N/A	[32, unisgned integer]
TFT	3	Descriptors Subtypes namespace.	Format per TS 24.008 Section 10.5.6.12.
IMSI	N/A	Context (new attribute)	[64, unsigned integer]
EBI	N/A	Context (new attribute)	[4, unsigned integer]
3GPP_QOS	4	QoS Subtypes namespace.	[8, qci] [32, gbr] [32, mbr] [32, apn_ambr] [32, ue_ambr] ARP
ARP	N/A	N/A	See Allocation-Retention-Priority from [RFC7222]

Table 22: 3GPP Attributes and Structures

The following COMMAND_SET values are supported for 3GPP.

- assign-ip - Assign the IP Address for the mobile session.
- assign-dpn - Assign the Dataplane Node.
- assign-fteid-ip - Assign the Fully Qualified TEID (F-TEID) LOCAL IP address.
- assign-fteid-teid - Assign the Fully Qualified TEID (F-TEID) LOCAL TEID.
- session - Assign values for the Session Level. When this involves 'assign-fteid-ip' and 'assign-fteid-teid' this implies the values are part of the default bearer.
- uplink - Command applies to uplink.
- downlink - Command applies to downlink.

8. Implementation Status

Two FPC Agent implementations have been made to date. The first was based upon Version 03 of the draft and followed Model 1. The second follows Version 04 of the document. Both implementations were OpenDaylight plug-ins developed in Java by Sprint. Version 03 was known as fpcagent and version 04's implementation is simply referred to as 'fpc'.

fpcagent's intent was to provide a proof of concept for FPC Version 03 Model 1 in January 2016 and research various errors, corrections and optimizations that the Agent could make when supporting multiple DPNs.

As the code developed to support OpenFlow and a proprietary DPN from a 3rd party, several of the advantages of a multi-DPN Agent became obvious including the use of machine learning to reduce the number of Flows and Policy entities placed on the DPN. This work has driven new efforts in the DIME WG, namely Diameter Policy Groups [I-D.bertz-dime-policygroups].

A throughput performance of tens per second using various NetConf based solutions in OpenDaylight made fpcagent undesirable for call processing. The RPC implementation improved throughput by an order of magnitude but was not useful based upon FPC's Version 03 design using two information models. During this time the features of version 04 and its converged model became attractive and the fpcagent project was closed in August 2016. fpcagent will no longer be developed and will remain a proprietary implementation.

The learnings of fpcagent has influenced the second project, fpc. Fpc is also an OpenDaylight project but is being prepared for open source release as the Opendaylight FpcAgent plugin (https://wiki.opendaylight.org/view/Project_Proposals:FpcAgent). This project is scoped to be a fully compliant FPC Agent that supports multiple DPNs including those that communicate via OpenFlow. The following features present in this draft and others developed by the FPC development team have already lead to an order of magnitude improvement.

Migration of non-realtime provisioning of entities such as topology and policy allowed the implementation to focus only on the rpc.

Using only 5 messages and 2 notifications has also reduced implementation time.

Command Sets, an optional feature in this specification, have eliminated 80% of the time spent determining what needs to be done with a Context during a Create or Update operation.

Op Reference is an optional feature modeled after video delivery. It has reduced unnecessary cache lookups. It also has the additional benefit of allowing an Agent to become cacheless and effectively act as a FPC protocol adapter remotely with multi-DPN support or colocated on the DPN in a single-DPN support model.

Multi-tenant support allows for Cache searches to be partitioned for clustering and performance improvements. This has not been capitalized upon by the current implementation but is part of the development roadmap.

Use of Contexts to pre-provision policy has also eliminated any processing of Ports for DPNs which permitted the code for CONFIGURE and CONF_BUNDLE to be implemented as a simple nested FOR loops (see below).

Current performance results without code optimizations or tuning allow 2-5K FPC Contexts processed per second on a 2013 Mac laptop. This results in 2x the number of transactions on the southbound interface to a proprietary DPN API on the same machine.

fpc currently supports the following:

1 proprietary DPN API

Policy and Topology as defined in this
specification using OpenDaylight North Bound
Interfaces such as NetConf and RestConf

CONFIG and CONF_BUNDLE (all operations)

DPN assignment, Tunnel allocations and IPv4
address assignment by the Agent or Client.

Immediate Response is always an
OK_NOTIFY_FOLLOWS.

```
     assignment system (receives rpc call):
       perform basic operation integrity check
       if CONFIG then
         goto assignments
         if assignments was ok then
           send request to activation system
           respond back to client with assignment data
         else
           send back error
         end if
       else if CONF_BUNDLE then
         for each operation in bundles
         goto assignments
         if assignments was ok then
           hold onto data
         else
           return error with the assignments that occurred in
             prior operations (best effort)
         end if
         end for
         send bundles to activation systems
       end if

     assignments:
       assign DPN, IPv4 Address and/or tunnel info as required
       if an error occurs undo all assignments in this operation
       return result

     activation system:
       build cache according to op-ref and operation type
       for each operation
         for each Context
           for each DPN / direction in Context
             perform actions on DPN according to Command Set
           end for
         end for
       end for
       commit changes to in memory cache
       log transaction for tracking and notification
                                     (CONFIG_RESULT_NOTIFY)
```

Figure 27: fpc pseudo code

For further information please contact Lyle Bertz who is also a co-author of this document.

NOTE: Tenant support requires binding a Client ID to a Tenant ID (it is a one to many relation) but that is outside of the scope of this specification. Otherwise, the specification is complete in terms of providing sufficient information to implement an Agent.

9. Security Considerations

Detailed protocol implementations for DMM Forwarding Policy Configuration must ensure integrity of the information exchanged between an FPC Client and an FPC Agent. Required Security Associations may be derived from co-located functions, which utilize the FPC Client and FPC Agent respectively.

The YANG modules defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

The information model defined in the memo is designed to be access by protocols specified in extensions to this document or, if using the YANG modules, as described above.

There are a number of data nodes defined which are writable/creatable/deletable. These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., a NETCONF edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/ vulnerability:

Nodes under the Policy tree provide generic policy enforcement and traffic classification. They can be used to block or permit traffic. If this portion of the model was to be compromised it may be used to block, identify or permit traffic that was not intended by the Tenant or FPC CLient.

Nodes under the Topology tree provide defintion of the Tenant’s forwarding topology. Any compromise of this information will provide topology information that could be used for subsequent attack vectors. Removal of topology can limit services.

Nodes under the Mobility Tree are runtime only and manipulated by remote procedure calls. The unwanted deletion or removal of such information would deny users service or provide services to unauthorized parties.

Some of the readable data nodes defined may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

IP address assignments in the Context along with their associated tunnel configurations/identifiers (from the FPC base module)

Internaional Mobile Subscriber Identity (IMSI) and bearer identifiers in the Context when using the optional 3GPP module

Some of the RPC operations defined may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

CONFIG and CONF_BUNDLE send Context information which can include information of a sensitive or vulnerable nature in some network environments as described above.

Monitor related RPC operations do not specicially provide sensitive or vulnerable informaiton but care must be taken by users to avoid identifier values that expose sensitive or vulnerable information.

Notications MUST be treated with same level of protection and scrutiny as the operations they correspond to. For example, a CONFIG_RESULT_NOTIFY notification provides the same information that is sent as part of the input and output of the CONFIG and CONF_BUNDLE RPC operations.

General usage of FPC MUST consider the following:

FPC Naming Section 4.4 permits arbirtrary string values but a users MUST avoid placing sensitive or vulnerable information in those values.

Policies that are very narrow and permit the identification of specific traffic, e.g. that of a single user, SHOULD be avoided.

10. IANA Considerations

This document registers six URIs in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations have been made.

URI: urn:ietf:params:xml:ns:yang:ietf-dmm-fpc
Registrant Contact: The DMM WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

```
      URI: urn:ietf:params:xml:ns:yang:ietf-dmm-threegpp
      Registrant Contact: The DMM WG of the IETF.
      XML: N/A, the requested URI is an XML namespace.

      URI: urn:ietf:params:xml:ns:yang:ietf-dmm-pmip-qos
      Registrant Contact: The DMM WG of the IETF.
      XML: N/A, the requested URI is an XML namespace.

      URI: urn:ietf:params:xml:ns:yang:ietf-dmm-traffic-selector-types
      Registrant Contact: The DMM WG of the IETF.
      XML: N/A, the requested URI is an XML namespace.

      URI: urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-policyext
      Registrant Contact: The DMM WG of the IETF.
      XML: N/A, the requested URI is an XML namespace.

      URI: urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-pmip
      Registrant Contact: The DMM WG of the IETF.
      XML: N/A, the requested URI is an XML namespace.
```

This document registers the following YANG modules in the "YANG Module Names" registry [RFC6020].

```
     name:         ietf-dmm-fpc
     namespace:    urn:ietf:params:xml:ns:yang:ietf-dmm-fpc
     prefix:       fpc
     reference:    TBD1

     name:         ietf-dmm-threegpp
     namespace:    urn:ietf:params:xml:ns:yang:ietf-dmm-threegpp
     prefix:       threegpp
     reference:    TBD1

     name:         ietf-dmm-pmip-qos
     namespace:    urn:ietf:params:xml:ns:yang:ietf-dmm-pmip-qos
     prefix:       qos-pmip
     reference:    TBD1

     name:         ietf-dmm-traffic-selector-types
     namespace:    urn:ietf:params:xml:ns:yang:
       ietf-dmm-traffic-selector-types
     prefix:       traffic-selectors
     reference:    TBD1

     name:         ietf-dmm-traffic-selector-types
     namespace:    urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-policyext
     prefix:       fpcpolicyext
     reference:    TBD1
     name:         ietf-dmm-traffic-selector-types
     namespace:    urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-pmip
     prefix:       fpc-pmip
     reference:    TBD1
```

The document registers the following YANG submodules in the "YANG Module Names" registry [RFC6020].

```
           name:         ietf-dmm-fpc-base
           parent:       ietf-dmm-fpc
           reference:    TBD1
```

11. Work Team Participants

Participants in the FPSM work team discussion include Satoru Matsushima, Danny Moses, Sri Gundavelli, Marco Liebsch, Pierrick Seite, Alper Yegin, Carlos Bernardos, Charles Perkins and Fred Templin.

12. References

12.1. Normative References

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-07 (work in progress), July 2017.

[I-D.ietf-sfc-nsh]
Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", draft-ietf-sfc-nsh-20 (work in progress), September 2017.

[I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-10 (work in progress), June 2017.

[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>.

[RFC6088] Tsirtsis, G., Giarreta, G., Soliman, H., and N. Montavont, "Traffic Selectors for Flow Bindings", RFC 6088, DOI 10.17487/RFC6088, January 2011, <https://www.rfc-editor.org/info/rfc6088>.

[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, DOI 10.17487/RFC6089, January 2011, <https://www.rfc-editor.org/info/rfc6089>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[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>.

12.2. Informative References

[I-D.bertz-dime-policygroups]
Bertz, L. and M. Bales, "Diameter Policy Groups and Sets", draft-bertz-dime-policygroups-04 (work in progress), June 2017.

[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.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-18 (work in progress), October 2016.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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>.

[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>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7222] Liebsch, M., Seite, P., Yokota, H., Korhonen, J., and S. Gundavelli, "Quality-of-Service Option for Proxy Mobile IPv6", RFC 7222, DOI 10.17487/RFC7222, May 2014, <https://www.rfc-editor.org/info/rfc7222>.

Appendix A. YANG Data Model for the FPC protocol

These modules define YANG definitions. Seven modules are defined:

- ietf-dmm-fpc (fpc) - Defines the base model and messages for FPC
- ietf-dmm-fpc-base An FPC submodule that defines the information model that is specified in this document
- ietf-pmip-qos (pmip-qos) - Defines proxy mobile IPv6 QoS parameters per RFC 7222
- ietf-traffic-selectors-types (traffic-selectors) - Defines Traffic Selectors per RFC 6088
- ietf-dmm-threegpp - Defines the base structures for 3GPP based IP mobility and augments fpcagent to support these parameters.
- ietf-dmm-fpc-pmip - Augments fpcp-base to include PMIP Traffic Selectors as a Traffic Descriptor subtype and pmip-qos QoS parameters, where applicable, as properties.
- ietf-dmm-fpc-policyext - defines basic policy extensions, e.g. Actions and Descriptors, to fpcbase and as defined in this document.

A.1. FPC Agent YANG Model

This module defines the information model and protocol elements specified in this document.

This module references [RFC6991] and the fpc-base module defined in this document.

```
   <CODE BEGINS> file "ietf-dmm-fpc@2017-03-08.yang"
   module ietf-dmm-fpc {
       namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc";
       prefix fpc;

       import ietf-inet-types { prefix inet; revision-date 2013-07-15; }

       include ietf-dmm-fpc-base;

       organization "IETF Distributed Mobility Management (DMM)
         Working Group";

       contact
          "WG Web:   <http://tools.ietf.org/wg/netmod/>
           WG List:  <mailto:netmod@ietf.org>

           WG Chair: Dapeng Liu
                     <mailto:maxpassion@gmail.com>

           WG Chair: Jouni Korhonen
                     <mailto:jouni.nospam@gmail.com>

           Editor:   Satoru Matsushima
                     <mailto:satoru.matsushima@g.softbank.co.jp>

           Editor:   Lyle Bertz
                     <mailto:lylebe551144@gmail.com>";

       description
       "This module contains YANG definition for
        Forwarding Policy Configuration Protocol (FPCP).

        Copyright (c) 2016 IETF Trust and the persons identified as the
        document authors. All rights reserved.

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

       revision 2017-03-08 {
           description "Version 06 updates.";
           reference "draft-ietf-dmm-fpc-cpdp-06";
       }

       revision 2016-08-03 {
           description "Initial Revision.";
           reference "draft-ietf-dmm-fpc-cpdp-05";
       }
       feature fpc-cloning {
         description "An ability to support cloning in the RPC.";
       }
       feature fpc-basename-registry {
         description "Ability to track Base Names already provisioned
           on the Agent";
       }
       feature fpc-bundles {
         description "Ability for Client to send multiple bundles of
           actions to an Agent";
       }
       feature fpc-client-binding {
         description "Allows a FPC Client to bind a DPN to an Topology
           Object";
       }
       feature fpc-auto-binding {
         description "Allows a FPC Agent to advertise Topology Objects
           that could be DPNs";
       }
       feature instruction-bitset {
         description "Allows the expression of instructions (bit sets)
           over FPC.";
       }
       feature operation-ref-scope {
         description "Provides the scope of refeneces in an operation.
           Used to optmize the Agent processing.";
       }
       feature policy-rpc-provisioning {
         description "Enables the ability to send policy elements
           (Policy Groups, Policies, Descriptors and Actions) to be sent
           in CONF or CONF_BUNDLE operations.";
       }

       typedef agent-identifier {
           type fpc:fpc-identity;
           description "Agent Identifier";
       }

       typedef client-identifier {
           type fpc:fpc-identity;
           description "Client Identifier";
       }
       grouping basename-info {
             leaf basename {
               if-feature fpc:fpc-basename-registry;
               type fpc:fpc-identity;
               description "Rules Basename";
             }
             leaf base-state {
               if-feature fpc:fpc-basename-registry;
               type string;
               description "Current State";
             }
             leaf base-checkpoint {
               if-feature fpc:fpc-basename-registry;
               type string;
               description "Checkpoint";
             }
             description "Basename Information";
       }

       // Top Level Structures
       container tenants {
           list tenant {
               key "tenant-id";
               leaf tenant-id {
                   type fpc:fpc-identity;
                   description "Tenant ID";
               }

               container fpc-policy {
                 list policy-groups {
                     key "policy-group-id";
                     uses fpc:fpc-policy-group;
                     description "Policy Groups";
                 }
                 list policies {
                     key "policy-id";
                     uses fpc:fpc-policy;
                     description "Policies";
                 }
                 list descriptors {
                   key descriptor-id;
                   uses fpc:fpc-descriptor;
                   description "Descriptors";
                 }
                 list actions {
                     key action-id;
                     uses fpc:fpc-action;
                     description "Actions";
                 }
                 description "Policy";
               }

               container fpc-mobility {
                 config false;
                 list contexts {
                     key context-id;
                     uses fpc:fpc-context;
                     description "Contexts";
                 }
                 list vports {
                     key vport-id;
                     uses fpc:fpc-vport;
                     description "Ports";
                 }
                 list monitors {
                     uses fpc:monitor-config;
                     description "Monitors";
                 }
                 description "Mobility";
               }
               container fpc-topology {
                 // Basic Agent Topology Structures
                 list domains {
                   key domain-id;
                   uses fpc:fpc-domain;
                   uses fpc:basename-info;
                   description "Domains";
                 }

                 leaf dpn-id {
                   if-feature fpc:fpc-basic-agent;
                   type fpc:fpc-dpn-id;
                   description "DPN ID";
                 }
                 leaf-list control-protocols {
                   if-feature fpc:fpc-basic-agent;
                   type identityref {
                     base "fpc:fpc-dpn-control-protocol";
                   }
                   description "Control Protocols";
                 }

                 list dpn-groups {
                     if-feature fpc:fpc-multi-dpn;
                     key dpn-group-id;
                     uses fpc:fpc-dpn-group;
                     list domains {
                       key domain-id;
                       uses fpc:fpc-domain;
                       uses fpc:basename-info;
                       description "Domains";
                     }
                     description "DPN Groups";
                 }
                 list dpns {
                     if-feature fpc:fpc-multi-dpn;
                     key dpn-id;
                     uses fpc:fpc-dpn;
                     description "DPNs";
                 }
                 description "Topology";
               }
           description "Tenant";
         }
         description "Tenant List";
     }

     container fpc-agent-info {
       // General Agent Structures
       leaf-list supported-features {
         type string;
         description "Agent Features";
       }

       // Common Agent Info
       list supported-events {
         key event;
         leaf event {
           type identityref {
             base "fpc:event-type";
           }
           description "Event Types";
         }
         leaf event-id {
           type fpc:event-type-id;
           description "Event ID";
         }
         description "Supported Events";
       }

       list supported-error-types {
         key error-type;
         leaf error-type {
           type identityref {
             base "fpc:error-type";
           }
           description "Error Type";
         }
         leaf error-type-id {
           type fpc:error-type-id;
           description "Error Type ID";
         }
         description "Supported Error Types";
       }
       description "General Agent Information";
     }


     // Multi-DPN Agent Structures
     grouping fpc-dpn-group {
         leaf dpn-group-id {
             type fpc:fpc-dpn-group-id;
             description "DPN Group ID";
         }
         leaf data-plane-role {
             type identityref {
                 base "fpc:fpc-data-plane-role";
             }
             description "Dataplane Role";
         }
         leaf access-type {
             type identityref {
                 base "fpc:fpc-access-type";
             }
             description "Access Type";
         }
         leaf mobility-profile {
             type identityref {
                 base "fpc:fpc-mobility-profile-type";
             }
             description "Mobility Profile";
         }
         list dpn-group-peers {
             key "remote-dpn-group-id";
             uses fpc:fpc-dpn-peer-group;
             description "Peer DPN Groups";
         }
         description "FPC DPN Group";
     }


     // RPC
```

```
       // RPC Specific Structures
       //Input Structures
       typedef admin-status {
           type enumeration {
               enum enabled {
                 value 0;
                 description "enabled";
               }
               enum disabled {
                 value 1;
                 description "disabled";
               }
               enum virtual {
                 value 2;
                 description "virtual";
               }
           }
           description "Adminstrative Status";
       }

       typedef session-status {
           type enumeration {
               enum complete {
                 value 0;
                 description "complete";
               }
               enum incomplete {
                 value 1;
                 description "incomplete";
               }
               enum outdated {
                 value 2;
                 description "outdated";
               }
           }
           description "Session Status";
       }

       typedef op-delay {
           type uint32;
           description "Operation Delay (ms)";
       }

       typedef op-identifier {
           type uint64;
           description "Operation Identifier";
       }
       typedef ref-scope {
         type enumeration {
           enum none {
             value 0;
             description "no references";
           }
           enum op {
             value 1;
             description "op - All references are contained in the
               operation body (intra-op)";
           }
           enum bundle {
             value 2;
             description "bundle - All references in exist in bundle
               (inter-operation/intra-bundle).
               NOTE - If this value comes in CONFIG call it is
                 equivalent to ’op’.";
           }
           enum storage {
             value 3;
             description "storage - One or more references exist outside
               of the operation and bundle. A lookup to a cache /
               storage is required.";
           }
           enum unknown {
             value 4;
             description " unknown - the location of the references are
               unknown.  This is treated as a ’storage’ type.";
           }
         }
         description "Search scope for references in the operation.";
       }

       grouping instructions {
         container instructions {
           if-feature instruction-bitset;
           choice instr-type {
             description "Instruction Value Choice";
           }
           description "Instructions";
         }
         description "Instructions Value";
       }

       grouping op-header {
         leaf client-id {
           type fpc:client-identifier;
           description "Client ID";
         }
         leaf delay {
           type op-delay;
           description "Delay";
         }
         leaf session-state {
           type session-status;
           description "Session State";
         }
         leaf admin-state {
           type admin-status;
           description "Admin State";
         }
         leaf op-type {
           type enumeration {
             enum create {
               value 0;
               description "create";
             }
             enum update {
               value 1;
               description "update";
             }
             enum query {
               value 2;
               description "query";
             }
             enum delete {
               value 3;
               description "delete";
             }
           }
           description "Type";
         }
         leaf op-ref-scope {
             if-feature operation-ref-scope;
             type fpc:ref-scope;
             description "Reference Scope";
         }
         uses fpc:instructions;
         description "Operation Header";
       }

       grouping clone-ref {
         leaf entity {
           type fpc:fpc-identity;
           description "Clone ID";
         }
         leaf source {
           type fpc:fpc-identity;
           description "Source";
         }
         description "Clone Reference";
       }

       identity command-set {
         description "protocol specific commands";
       }

       grouping context-operation {
         uses fpc:fpc-context;
         uses fpc:instructions;
         description "Context Operation";
       }

       // Output Structure
       grouping payload {
         list ports {
           uses fpc:fpc-vport;
           description "Ports";
         }
         list contexts {
           uses fpc:context-operation;
           description "Contexts";
         }
         list policy-groups {
           if-feature fpc:policy-rpc-provisioning;
           key "policy-group-id";
           uses fpc:fpc-policy-group;
           description "Policy Groups";
         }
         list policies {
           if-feature fpc:policy-rpc-provisioning;
           key "policy-id";
           uses fpc:fpc-policy;
           description "Policies";
         }
         list descriptors {
           if-feature fpc:policy-rpc-provisioning;
           key descriptor-id;
           uses fpc:fpc-descriptor;
           description "Descriptors";
         }
         list actions {
           if-feature fpc:policy-rpc-provisioning;
           key action-id;
```

```
             uses fpc:fpc-action;
             description "Actions";
           }
           description "Payload";
         }

         grouping op-input {
           uses fpc:op-header;
           leaf op-id {
             type op-identifier;
             description "Operation ID";
           }
           choice op_body {
             case create_or_update {
               list clones {
                 if-feature fpc-cloning;
                 key entity;
                 uses fpc:clone-ref;
                 description "Clones";
               }
               uses fpc:payload;
               description "Create/Update input";
             }
             case delete_or_query {
               uses fpc:targets-value;
               description "Delete/Query input";
             }
             description "Opeartion Input value";
           }
           description "Operation Input";
         }

         typedef result {
           type enumeration {
             enum ok {
               value 0;
               description "OK";
             }
             enum err {
               value 1;
               description "Error";
             }
             enum ok-notify-follows {
               value 2;
               description "OK with NOTIFY following";
             }
           }
           description "Result Status";
```

```
       }

       identity error-type {
         description "Base Error Type";
       }
       identity name-already-exists {
         description "Notification that an entity of the same name
           already exists";
       }

       typedef error-type-id {
         type uint32;
         description "Integer form of the Error Type";
       }

       grouping op-status-value {
         leaf op-status {
           type enumeration {
             enum ok {
               value 0;
               description "OK";
             }
             enum err {
               value 1;
               description "Error";
             }
           }
           description "Operation Status";
         }
         description "Operation Status Value";
       }

       grouping error-info {
             leaf error-type-id {
               type fpc:error-type-id;
               description "Error ID";
             }
             leaf error-info {
               type string {
                 length "1..1024";
               }
               description "Error Detail";
             }
             description "Error Information";
       }

       grouping result-body {
         leaf op-id {
           type op-identifier;
           description "Operation Identifier";
         }
         choice result-type {
           case err {
             uses fpc:error-info;
             description "Error Information";
           }
           case create-or-update-success {
             uses fpc:payload;
             description "Create/Update Success";
           }
           case delete_or_query-success {
             uses fpc:targets-value;
             description "Delete/Query Success";
           }
           case empty-case {
             description "Empty Case";
           }
           description "Result Value";
         }
         description "Result Body";
       }

       // Common RPCs
       rpc configure {
         description "CONF message";
         input {
           uses fpc:op-input;
         }
         output {
           leaf result {
             type result;
             description "Result";
           }
           uses fpc:result-body;
         }
       }

       rpc configure-bundles {
         if-feature fpc:fpc-bundles;
         description "CONF_BUNDLES message";
         input {
           leaf highest-op-ref-scope {
               if-feature operation-ref-scope;
               type fpc:ref-scope;
               description "Highest Op-Ref used in the input";
           }
           list bundles {
             key op-id;
             uses fpc:op-input;
             description "List of operations";
           }
         }
         output {
           list bundles {
             key op-id;
             uses fpc:result-body;
             description "Operation Identifier";
           }
         }
       }

       // Notification Messages & Structures
       typedef notification-id {
         type uint32;
         description "Notification Identifier";
       }

       grouping notification-header {
         leaf notification-id {
             type fpc:notification-id;
             description "Notification ID";
         }
         leaf timestamp {
             type uint32;
             description "timestamp";
         }
         description "Notification Header";
       }

       notification config-result-notification {
         uses fpc:notification-header;
         choice value {
           case config-result {
             uses fpc:op-status-value;
             uses fpc:result-body;
             description "CONF Result";
           }
           case config-bundle-result {
             list bundles {
               uses fpc:op-status-value;
               uses fpc:result-body;
               description "Operation Results";
             }
             description "CONF_BUNDLES Result";
```

```
          }
          description "Config Result value";
        }
        description "CONF/CONF_BUNDLES Async Result";
      }

      rpc event_register {
        description "Used to register monitoring of parameters/events";
          input {
            uses fpc:monitor-config;
          }
          output {
            leaf monitor-result {
              type fpc:result;
              description "Result";
            }
            uses fpc:error-info;
          }
      }

      rpc event_deregister {
        description "Used to de-register monitoring of
            parameters/events";
          input {
            list monitors {
              uses fpc:monitor-id;
              description "Monitor ID";
            }
          }
          output {
            leaf monitor-result {
              type fpc:result;
              description "Result";
            }
            uses fpc:error-info;
          }

      }

      rpc probe {
          description "Probe the status of a registered monitor";
          input {
            uses fpc:targets-value;
          }
          output {
            leaf monitor-result {
              type fpc:result;
              description "Result";
```

```
             }
             uses fpc:error-info;
           }
       }

       notification notify {
           uses fpc:notification-header;
           choice value {
               case dpn-candidate-available {
                 if-feature fpc:fpc-auto-binding;
                 leaf node-id {
                   type inet:uri;
                   description "Topology URI";
                 }
                 leaf-list access-types {
                   type identityref {
                     base "fpc:fpc-access-type";
                   }
                   description "Access Types";
                 }
                 leaf-list mobility-profiles {
                   type identityref {
                     base "fpc:fpc-mobility-profile-type";
                   }
                   description "Mobility Profiles";
                 }
                 leaf-list forwarding-plane-roles {
                   type identityref {
                     base "fpc:fpc-data-plane-role";
                   }
                   description "Forwarding Plane Role";
                 }
                 description "DPN Candidate Availability";
               }
               case monitor-notification {
                 choice monitor-notification-value {
                   case monitoring-suspension {
                     leaf monitoring-suspended {
                       type empty;
                       description "Indicates that monitoring has
                         uspended";
                     }
                     leaf suspension-note {
                       type string;
                       description "Indicates the monitoring
                         suspension reason";
                     }
                   }
                   case monitoring-resumption {
                     leaf monitoring-resumed {
                       type empty;
                       description "Indicates that monitoring
                         has resumed";
                     }
                   }
                   case simple-monitor {
                     uses fpc:report;
                     description "Report";
                   }
                   case bulk-monitors {
                     list reports {
                       uses fpc:report;
                       description "Reports";
                     }
                     description "Bulk Monitor Response";
                   }
                   description "Monitor Notification value";
                 }
                 description "Monitor Notification";
               }
               description "Notify Value";
           }
           description "Notify Message";
       }
   }
   <CODE ENDS>
```

A.2. YANG Models

A.2.1. FPC YANG Model

This module defines the base data elements specified in this document.

This module references [RFC6991].

```
   <CODE BEGINS> file "ietf-dmm-fpc-base@2017-03-08.yang"
   submodule ietf-dmm-fpc-base {
       belongs-to ietf-dmm-fpc {
          prefix fpc;
       }

       import ietf-inet-types { prefix inet; revision-date 2013-07-15; }
       import ietf-yang-types { prefix ytypes;
           revision-date 2013-07-15; }
       organization "IETF Distributed Mobility Management (DMM)
         Working Group";

       contact
          "WG Web:   <http://tools.ietf.org/wg/netmod/>
           WG List:  <mailto:netmod@ietf.org>

           WG Chair: Dapeng Liu
                     <mailto:maxpassion@gmail.com>

           WG Chair: Jouni Korhonen
                     <mailto:jouni.nospam@gmail.com>

           Editor:   Satoru Matsushima
                     <mailto:satoru.matsushima@g.softbank.co.jp>

           Editor:   Lyle Bertz
                     <mailto:lylebe551144@gmail.com>";

       description
       "This module contains YANG definition for
        Forwarding Policy Configuration Protocol(FPCP).

        Copyright (c) 2016 IETF Trust and the persons identified as the
        document authors. All rights reserved.

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

       revision 2017-03-08 {
           description "Version 06 updates.";
           reference "draft-ietf-dmm-fpc-cpdp-06";
       }

       revision 2016-08-03 {
           description "Initial Revision.";
           reference "draft-ietf-dmm-fpc-cpdp-05";
       }

       feature fpc-basic-agent {
           description "This is an agent co-located with a DPN.  In this
           case only DPN Peer Groups, the DPN Id and Control Protocols
           are exposed along with the core structures.";
       }
       feature fpc-multi-dpn {
           description "The agent supports multiple DPNs.";
       }

       typedef fpc-identity {
           type union {
               type uint32;
               type string;
               type instance-identifier;
           }
           description "FPC Identity";
       }

       grouping target-value {
         leaf target {
             type fpc-identity;
             description "Target Identity";
         }
         description "FPC Target Value";
       }

       grouping targets-value {
         list targets {
             key "target";
             leaf target {
               type fpc-identity;
               description "Target Id";
             }
             leaf dpn-id {
                   type fpc:fpc-dpn-id;
                   description "DPN Id";
             }
             description "List of Targets";
         }
         description "Targets Value";
       }

       // Descriptor Structure
       typedef fpc-descriptor-id-type {
           type fpc:fpc-identity;
           description "Descriptor-ID";
       }
       identity fpc-descriptor-type {
           description "A traffic descriptor";
       }
       grouping fpc-descriptor-id {
         leaf descriptor-id {
           type fpc:fpc-identity;
           description "Descriptor Id";
         }
         description "FPC Descriptor ID value";
       }
       grouping fpc-descriptor {
           uses fpc:fpc-descriptor-id;
           leaf descriptor-type {
             type identityref {
               base "fpc-descriptor-type";
             }
             mandatory true;
             description "Descriptor Type";
           }
           choice descriptor-value {
             case all-traffic {
               leaf all-traffic {
                 type empty;
                 description "Empty Value";
               }
             }
             description "Descriptor Value";
           }
           description "FPC Descriptor";
       }

       // Action Structure
       typedef fpc-action-id-type {
           type fpc:fpc-identity;
           description "Action-ID";
       }
       identity fpc-action-type {
           description "Action Type";
       }
       grouping fpc-action-id {
         leaf action-id {
           type fpc:fpc-action-id-type;
           description "Action Identifier";
         }
         description "FPC Action ID";
       }
       grouping fpc-action {
           uses fpc:fpc-action-id;
           leaf action-type {
             type identityref {
               base "fpc-action-type";
             }
             mandatory true;
             description "Action Type";
           }
           choice action-value {
             case drop {
               leaf drop {
                 type empty;
                 description "Empty Value";
               }
             }
             description "FPC Action Value";
           }
           description "FPC Action";
       }

       // Rule Structure
       grouping fpc-rule {
           list descriptors {
             key descriptor-id;
             uses fpc:fpc-descriptor-id;
             leaf direction {
               type fpc:fpc-direction;
               description "Direction";
             }
             description "Descriptors";
           }
           list actions {
             key action-id;
             leaf action-order {
                 type uint32;
                 description "Action Execution Order";
             }
             uses fpc:fpc-action-id;
             description "Actions";
           }
           description
             "FPC Rule.  When no actions are present the action is DROP.
             When no Descriptors are empty the default is
             'all traffic'.";
       }

       // Policy Structures
       typedef fpc-policy-id {
           type fpc:fpc-identity;
           description "Policy Identifier";
       }
       grouping fpc-policy {
           leaf policy-id {
               type fpc:fpc-policy-id;
               description "Policy Id";
           }
           list rules {
               key order;
               leaf order {
                 type uint32;
                 description "Rule Order";
               }
               uses fpc:fpc-rule;
               description "Rules";
           }
           description "FPC Policy";
       }

       // Policy Group
       typedef fpc-policy-group-id {
           type fpc:fpc-identity;
           description "Policy Group Identifier";
       }
       grouping fpc-policy-group {
         leaf policy-group-id {
           type fpc:fpc-policy-group-id;
           description "Policy Group ID";
         }
         leaf-list policies {
           type fpc:fpc-policy-id;
           description "Policies";
         }
         description "FPC Policy Group";
       }

       // Mobility Structures
       // Port Group
       typedef fpc-vport-id {
           type fpc:fpc-identity;
           description "FPC Port Identifier";
       }
       grouping fpc-vport {
           leaf vport-id {
               type fpc:fpc-vport-id;
               description "Port ID";
           }
           leaf-list policy-groups {
               type fpc:fpc-policy-group-id;
               description "Policy Groups";
           }
           description "FPC Port";
       }

       // Context Group
       typedef fpc-context-id {
           type fpc:fpc-identity;
           description "FPC Context Identifier";
       }
       grouping fpc-context-profile {
           leaf tunnel-local-address {
               type inet:ip-address;
               description "endpoint address of the DPN which a
                 gent exists.";
           }
           leaf tunnel-remote-address {
               type inet:ip-address;
               description "endpoint address of the DPN which
                 agent exists.";
           }
           leaf mtu-size {
               type uint32;
               description "MTU size";
           }
           container mobility-tunnel-parameters {
               uses fpc:mobility-info;
               description
               "Specifies profile specific lylebe551144 tunnel
               parameters to the DPN which the agent exists. The
               profiles includes GTP/TEID for 3gpp profile, GRE/Key for
               ietf-pmip profile, or new profile if anyone will define
               it.";
           }
           container nexthop {
               uses fpc:fpc-nexthop;
               description "Next Hop";
           }
           container qos-profile-parameters {
               uses fpc:fpc-qos-profile;
               description "QoS Parameters";
           }
           container dpn-parameters {
               description "DPN Parameters";
           }
           list vendor-parameters {
               key "vendor-id vendor-type";
               uses fpc:vendor-attributes;
               description "Vendor Parameters";
           }
```

```
           description "A profile that applies to a specific direction";
       }

       typedef fpc-direction {
          type enumeration {
            enum lylebe551144 {
              description "lylebe551144";
            }
            enum downlink {
              description "Downlink";
            }
            enum both {
              description "Both";
           }
          }
          description "FPC Direction";
       }

       grouping fpc-context {
           leaf context-id {
               type fpc:fpc-context-id;
               description "Context ID";
           }
           leaf-list vports {
               type fpc:fpc-vport-id;
               description "Vports";
           }
           leaf dpn-group {
             type fpc:fpc-dpn-group-id;
             description "DPN Group";
           }
           leaf-list delegated-ip-prefixes {
               type inet:ip-prefix;
               description "Delegated Prefix(es)";
           }
           container ul {
               if-feature fpc:fpc-basic-agent;
               uses fpc:fpc-context-profile;
               description "lylebe551144";
           }
           container dl {
               if-feature fpc:fpc-basic-agent;
               uses fpc:fpc-context-profile;
               description "Downlink";
           }
           list dpns {
               if-feature fpc:fpc-multi-dpn;
               key "dpn-id direction";
               leaf dpn-id {
                   type fpc:fpc-dpn-id;
                   description "DPN";
               }
               leaf direction {
                   type fpc:fpc-direction;
                   mandatory true;
                   description "Direction";
               }
               uses fpc:fpc-context-profile;
               description "DPNs";
           }
           leaf parent-context {
               type fpc:fpc-context-id;
               description "Parent Context";
           }
           description "FCP Context";
       }

       // Mobility (Tunnel) Information
       grouping mobility-info {
           choice profile-parameters {
               case nothing {
                 leaf none {
                   type empty;
                   description "Empty Value";
                 }
                 description "No Parameters Case";
               }
               description "Mobility Profile Parameters";
           }
           description "Mobility Information";
       }

       // Next Hop Structures
       typedef fpc-service-path-id {
           type uint32 {
               range "0..33554431";
           }
           description "SERVICE_PATH_ID";
       }
       typedef fpc-mpls-label {
           type uint32 {
             range "0..1048575";
           }
           description "MPLS label";
       }
       identity fpc-nexthop-type {
           description "Next Hop Type";
       }
       identity fpc-nexthop-ip {
           base "fpc:fpc-nexthop-type";
           description "Nexthop IP";
       }
       identity fpc-nexthop-servicepath {
           base "fpc:fpc-nexthop-type";
           description "Nexthop Service Path";
       }
       identity fpc-nexthop-mac {
           base "fpc:fpc-nexthop-type";
           description "Nexthop MAC-Address";
       }
       identity fpc-nexthop-mpls {
           base "fpc:fpc-nexthop-type";
           description "Nexthop MPLS";
       }
       identity fpc-nexthop-if {
           base "fpc:fpc-nexthop-type";
           description "Nexthop If index";
       }
       grouping fpc-nexthop {
           leaf nexthop-type {
               type identityref {
                 base "fpc:fpc-nexthop-type";
               }
               description "Nexthop Type";
           }
           choice nexthop-value {
               case ip-nexthop {
                   leaf ip {
                     type inet:ip-address;
                     description "IP Value";
                   }
                   description "IP Case";
               }
               case macaddress-nexthop {
                   leaf macaddress {
                     type ytypes:mac-address;
                     description "MAC Address Value";
                   }
               }
               case servicepath-nexthop {
                   leaf servicepath {
                       type fpc:fpc-service-path-id;
                       description "Service Path Value";
                   }
                   description "Service Path Case";
               }
               case mplslabel-nexthop {
                   leaf lsp {
                       type fpc:fpc-mpls-label;
                       description "MPLS Value";
                   }
                   description "Service Path Case";
               }
               case if-nexthop {
                   leaf if-index {
                       type uint16;
                       description "If (interface) Value";
                   }
                   description "Service Path Case";
               }
               description "Value";
           }
           description "Nexthop Value";
       }

       // QoS Information
       identity fpc-qos-type {
           description "Base identity from which specific uses of QoS
             types are derived.";
       }
       grouping fpc-qos-profile {
           leaf qos-type {
               type identityref {
                   base fpc:fpc-qos-type;
               }
               description "the profile type";
           }
           choice value {
               description "QoS Value";
           }
           description "QoS Profile";
       }

       // Vendor Specific Attributes
       identity vendor-specific-type {
           description "Vendor Specific Attribute Type";
       }
       grouping vendor-attributes {
           leaf vendor-id {
               type fpc:fpc-identity;
               description "Vendor ID";
```

```
            }
            leaf vendor-type {
                type identityref {
                    base "fpc:vendor-specific-type";
                }
                description "Attribute Type";
            }
            choice value {
                case empty-type {
                    leaf empty-type {
                        type empty;
                        description "Empty Value";
                    }
                    description "Empty Case";
                }
                description "Atttribute Value";
            }
            description "Vendor Specific Attributes";
        }

        // Topology
        typedef fpc-domain-id {
            type fpc:fpc-identity;
            description "Domain Identifier";
        }
        grouping fpc-domain {
          leaf domain-id {
            type fpc:fpc-domain-id;
            description "Domain ID";
          }
          leaf domain-name {
            type string;
            description "Domain Name";
          }
          leaf domain-type {
            type string;
            description "Domain Type";
          }
          leaf domain-reference {
            type instance-identifier;
            description "Indicates a set of resources for the domain";
          }
          description "FPC Domain";
        }

        typedef fpc-dpn-id {
            type fpc:fpc-identity;
            description "DPN Identifier";
```

```
       }
       identity fpc-dpn-control-protocol {
           description "DPN Control Protocol";
       }
       grouping fpc-dpn {
           leaf dpn-id {
             type fpc:fpc-dpn-id;
             description "DPN ID";
           }
           leaf dpn-name {
             type string;
             description "DPN Name";
           }
           leaf-list dpn-groups {
             type fpc:fpc-dpn-group-id;
             description "DPN Groups";
           }
           leaf node-reference {
             type instance-identifier;
             description "DPN => Node (Topology) Mapping";
           }
           description "FPC DPN";
       }

       typedef fpc-dpn-group-id {
           type fpc:fpc-identity;
           description "DPN Group Identifier";
       }
       identity fpc-data-plane-role {
           description "Role of DPN Group in the Forwarding Plane";
       }
       identity fpc-access-dpn-role {
          base "fpc:fpc-data-plane-role";
          description "Access DPN Role";
       }
       identity fpc-anchor-dpn-role {
          base "fpc:fpc-data-plane-role";
          description "Anchor DPN Role";
       }

       identity fpc-access-type {
           description "Access Type of the DPN Group";
       }
       identity fpc-mobility-profile-type {
           description "Mobility Profile Type";
       }

       grouping fpc-dpn-peer-group {
           leaf remote-dpn-group-id {
               type fpc:fpc-dpn-group-id;
               description "Remote DPN Group ID";
           }
           leaf remote-mobility-profile {
               type identityref {
                   base "fpc:fpc-mobility-profile-type";
               }
               description "Mobility Profile";
           }
           leaf remote-data-plane-role {
               type identityref {
                   base "fpc:fpc-data-plane-role";
               }
               description "Forwarding Plane Role";
           }
           leaf remote-endpoint-address {
               type inet:ip-address;
               description "Remote Endpoint Address";
           }
           leaf local-endpoint-address {
               type inet:ip-address;
               description "Local Endpoint Address";
           }
           leaf mtu-size {
               type uint32;
               description "MTU Size";
           }
           description "FPC DPN Peer Group";
       }

       // Events, Probes & Notifications
       identity event-type {
           description "Base Event Type";
       }
       typedef event-type-id {
           type uint32;
           description "Event ID Type";
       }

       grouping monitor-id {
         leaf monitor-id {
           type fpc:fpc-identity;
           description "Monitor Identifier";
         }
         description "Monitor ID";
       }
       identity report-type {
         description "Type of Report";
       }
       identity periodic-report {
         base "fpc:report-type";
         description "Periodic Report";
       }
       identity threshold-report {
         base "fpc:report-type";
         description "Threshold Report";
       }
       identity scheduled-report {
         base "fpc:report-type";
         description "Scheduled Report";
       }
       identity events-report {
         base "fpc:report-type";
         description "Events Report";
       }

       grouping report-config {
         choice event-config-value {
           case periodic-config {
               leaf period {
                 type uint32;
                 description "Period";
               }
               description "Periodic Config Case";
           }
           case threshold-config {
               leaf lo-thresh {
                 type uint32;
                 description "lo threshold";
               }
               leaf hi-thresh {
                 type uint32;
                 description "hi threshold";
               }
               description "Threshold Config Case";
           }
           case scheduled-config {
               leaf report-time {
                 type uint32;
                 description "Reporting Time";
               }
               description "Scheduled Config Case";
           }
           case events-config-ident {
               leaf-list event-identities {
                 type identityref {
                   base "fpc:event-type";
                 }
                 description "Event Identities";
               }
               description "Events Config Identities Case";
           }
           case events-config {
               leaf-list event-ids {
                 type uint32;
                 description "Event IDs";
               }
               description "Events Config Case";
           }
           description "Event Config Value";
         }
         description "Report Configuration";
       }

       grouping monitor-config {
         uses fpc:monitor-id;
         uses fpc:target-value;
         uses fpc:report-config;
         description "Monitor Configuration";
       }

       grouping report {
         uses fpc:monitor-config;
         choice report-value {
           leaf trigger {
             type fpc:event-type-id;
             description "Trigger Identifier";
           }
           case simple-empty {
             leaf nothing {
               type empty;
               description "Empty Value";
             }
             description "Empty Case";
           }
           case simple-val32 {
             leaf val32 {
               type uint32;
               description "Unsigned 32 bit value";
             }
             description "Simple Value Case";
           }
           description "Report Value";
         }
         description "Monitor Report";
       }
   }
   <CODE ENDS>
```

A.2.2. PMIP QoS Model

This module defines the base protocol elements specified in this document.

This module references [RFC6991] and the traffic-selector-types module defined in this document.

```
<CODE BEGINS> file "ietf-pmip-qos@2016-02-10.yang"
module ietf-pmip-qos {
    yang-version 1;

    namespace
      "urn:ietf:params:xml:ns:yang:ietf-pmip-qos";

    prefix "qos-pmip";

    import ietf-inet-types {
        prefix inet;
        revision-date 2013-07-15;
    }
    import ietf-traffic-selector-types { prefix traffic-selectors; }

    organization "IETF Distributed Mobility Management (DMM)
      Working Group";

    contact
       "WG Web:   <http://tools.ietf.org/wg/netmod/>
        WG List:  <mailto:netmod@ietf.org>

        WG Chair: Dapeng Liu
                  <mailto:maxpassion@gmail.com>

        WG Chair: Jouni Korhonen
                  <mailto:jouni.nospam@gmail.com>

        Editor:   Satoru Matsushima
                  <mailto:satoru.matsushima@g.softbank.co.jp>

        Editor:   Lyle Bertz
                  <mailto:lylebe551144@gmail.com>";
    description
      "This module contains a collection of YANG definitions for
     quality of service paramaters used in Proxy Mobile IPv6.

     Copyright (c) 2016 IETF Trust and the persons identified as the
     document authors. All rights reserved.

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

    revision 2016-02-10 {
        description "Initial revision";
        reference
         "RFC 7222: Quality-of-Service Option for Proxy Mobile IPv6";
    }

    // Type Definitions

    // QoS Option Field Type Definitions
    typedef sr-id {
        type uint8;
        description
         "An 8-bit unsigned integer used]
        for identifying the QoS Service Request.  Its uniqueness is
        within the scope of a mobility session.  The local mobility
        anchor always allocates the Service Request Identifier.
        When a new QoS Service Request is initiated by a mobile
        access gateway, the Service Request Identifier in the initial
        request message is set to a value of (0), and the local
        mobility anchor allocates a Service Request Identifier and
        includes it in the response.  For any new QoS Service
        Requests initiated by a local mobility anchor, the
        Service Request Identifier is set to the allocated value.";
    }

    typedef traffic-class {
        type inet:dscp;
        description
        "Traffic Class consists of a 6-bit DSCP field followed by a
         2-bit reserved field.";
       reference
```

```
         "RFC 3289: Management Information Base for the Differentiated
                    Services Architecture
          RFC 2474: Definition of the Differentiated Services Field
                    (DS Field) in the IPv4 and IPv6 Headers
          RFC 2780: IANA Allocation Guidelines For Values In
                    the Internet Protocol and Related Headers";
    }

    typedef operational-code {
        type enumeration {
            enum RESPONSE {
          value 0;
          description "Response to a QoS request";
        }
            enum ALLOCATE {
          value 1;
          description "Request to allocate QoS resources";
        }
            enum DE-ALLOCATE {
          value 2;
          description "Request to de-Allocate QoS resources";
        }
            enum MODIFY {
          value 3;
          description "Request to modify QoS parameters for a
              previously negotiated QoS Service Request";
        }
            enum QUERY {
          value 4;
          description "Query to list the previously negotiated QoS
              Service Requests that are still active";
        }
            enum NEGOTIATE {
          value 5;
          description "Response to a QoS Service Request with a
            counter QoS proposal";
        }
        }
        description
       "1-octet Operational code indicates the type of QoS request.
           Reserved values:   (6) to (255)
                Currently not used.  Receiver MUST ignore the option
                received with any value in this range.";
    }

    // QoS Attribute Types

    //The enumeration value for mapping - don’t confuse with the
    //  identities
    typedef qos-attrubite-type-enum {
        type enumeration {
            enum Reserved {
          value 0;
          description "This value is reserved and cannot be used";
        }
            enum Per-MN-Agg-Max-DL-Bit-Rate {
          value 1;
          description "Per-Mobile-Node Aggregate Maximum Downlink
              Bit Rate.";
        }
        enum Per-MN-Agg-Max-UL-Bit-Rate {
          value 2;
          description "Per-Mobile-Node Aggregate Maximum Uplink Bit
            Rate.";
        }
        enum Per-Session-Agg-Max-DL-Bit-Rate {
          value 3;
          description "Per-Mobility-Session Aggregate Maximum
            Downlink Bit Rate.";
        }
        enum Per-Session-Agg-Max-UL-Bit-Rate {
          value 4;
          description "Per-Mobility-Session Aggregate Maximum
             Uplink Bit Rate.";
        }
        enum Allocation-Retention-Priority {
          value 5;
          description "Allocation and Retention Priority.";
        }
        enum Aggregate-Max-DL-Bit-Rate {
          value 6;
          description "Aggregate Maximum Downlink Bit Rate.";
        }
        enum Aggregate-Max-UL-Bit-Rate {
          value 7;
          description "Aggregate Maximum Uplink Bit Rate.";
        }
        enum Guaranteed-DL-Bit-Rate {
          value 8;
          description "Guaranteed Downlink Bit Rate.";
        }
        enum Guaranteed-UL-Bit-Rate {
          value 9;
          description "Guaranteed Uplink Bit Rate.";
        }
        enum QoS-Traffic-Selector {
          value 10;
          description "QoS Traffic Selector.";
        }
        enum QoS-Vendor-Specific-Attribute {
          value 11;
          description "QoS Vendor-Specific Attribute.";
        }
        }
        description
        "8-bit unsigned integer indicating the type of the QoS
          attribute.  This specification reserves the following
        reserved values.
          (12) to (254) -  Reserved
             These values are reserved for future allocation.

          (255)  Reserved
             This value is reserved and cannot be used.";
    }

    // Attribute Type as Identities
    // Added for convenience of inclusion and extension in
    //    other YANG modules.
    identity qos-attribute-type {
        description
            "Base type for Quality of Service Attributes";
    }

    identity Per-MN-Agg-Max-DL-Bit-Rate-type {
        base qos-attribute-type;
        description
            "Per-Mobile-Node Aggregate Maximum Downlink Bit Rate.";
      }

    identity Per-MN-Agg-Max-UL-Bit-Rate-type {
        base qos-attribute-type;
        description
            "Per-Mobile-Node Aggregate Maximum Uplink Bit Rate";
    }

    identity Per-Session-Agg-Max-DL-Bit-Rate-type {
        base qos-attribute-type;
        description
       "Per-Mobility-Session Aggregate Maximum Downlink Bit Rate.";
  }

    identity Per-Session-Agg-Max-UL-Bit-Rate-type {
        base qos-attribute-type;
        description
```

```
          "Per-Mobility-Session Aggregate Maximum Uplink Bit Rate.";
      }

      identity Allocation-Retention-Priority-type {
          base qos-attribute-type;
          description
              "Allocation and Retention Priority.";
      }

      identity Aggregate-Max-DL-Bit-Rate-type {
          base qos-attribute-type;
          description "Aggregate Maximum Downlink Bit Rate.";
      }

    identity Aggregate-Max-UL-Bit-Rate-type {
        base qos-attribute-type;
        description "Aggregate Maximum Uplink Bit Rate.";
    }

    identity Guaranteed-DL-Bit-Rate-type {
        base qos-attribute-type;
        description "Guaranteed Downlink Bit Rate.";
    }

    identity Guaranteed-UL-Bit-Rate-type {
        base qos-attribute-type;
        description "Guaranteed Uplink Bit Rate.";
    }

    identity QoS-Traffic-Selector-type {
        base qos-attribute-type;
        description "QoS Traffic Selector.";
    }

    identity QoS-Vendor-Specific-Attribute-type {
        base qos-attribute-type;
        description "QoS Vendor-Specific Attribute.";
    }

    //value definitions
    typedef Per-MN-Agg-Max-DL-Bit-Rate-Value {
        type uint32;
        description
            "This is a 32-bit unsigned integer that
            indicates the aggregate maximum downlink bit rate that is
            requested/allocated for all the mobile node’s IP flows.
            The measurement units for Per-MN-Agg-Max-DL-Bit-Rate are
            bits per second.";
    }

    typedef Per-MN-Agg-Max-UL-Bit-Rate-Value {
        type uint32;
        description
            "This is a 32-bit unsigned integer that
              indicates the aggregate maximum uplink bit rate that is
            requested/allocated for the mobile node's IP flows.  The
            measurement units for Per-MN-Agg-Max-UL-Bit-Rate are bits
            per second.";
    }

    // Generic Structure for the uplink and downlink
    grouping Per-Session-Agg-Max-Bit-Rate-Value {
        leaf max-rate {
            type uint32;
            mandatory true;
            description
       "This is a 32-bit unsigned integer
        that indicates the aggregate maximum bit rate that is
        requested/allocated for all the IP flows associated with
        that mobility session.  The measurement units for
        Per-Session-Agg-Max-UL/DL-Bit-Rate are bits per second.";
        }
        leaf service-flag {
            type boolean;
            mandatory true;
            description
       "This flag is used for extending the scope of the
            target flows for Per-Session-Agg-Max-UL/DL-Bit-Rate
            from(UL)/to(DL) the mobile node's other mobility sessions
            sharing the same Service Identifier. 3GPP Access Point Name
            (APN) is an example of a Service Identifier, and that
            identifier is carried using the Service Selection mobility
            option [RFC5149].

            -  When the (S) flag is set to a value of (1), then the
               Per-Session-Agg-Max-Bit-Rate is measured as an
               aggregate across all the mobile node's other mobility
               sessions sharing the same Service Identifier associated
               with this mobility session.

            -  When the (S) flag is set to a value of (0), then the
               target flows are limited to the current mobility
               session.

            -  The (S) flag MUST NOT be set to a value of (1) when there
               is no Service Identifier associated with the mobility
```

```
                session.";
             reference
         "RFC 5149 - Service Selection mobility option";
         }
         leaf exclude-flag {
             type boolean;
             mandatory true;
             description
         "This flag is used to request that the uplink/downlink
                   flows for which the network is providing
             Guaranteed-Bit-Rate service be excluded from the
             target IP flows for which
             Per-Session-Agg-Max-UL/DL-Bit-Rate is measured.

             -  When the (E) flag is set to a value of (1), then the
                request is to exclude the IP flows for which
                Guaranteed-UL/DL-Bit-Rate is negotiated from the flows
                for which Per-Session-Agg-Max-UL/DL-Bit-Rate
                    is measured.

             -  When the (E) flag is set to a value of (0), then the
                request is not to exclude any IP flows from the target
                IP flows for which Per-Session-Agg-Max-UL/DL-Bit-Rate
                is measured.

             -  When the (S) flag and (E) flag are both set to a value
                of (1), then the request is to exclude all the IP flows
                sharing the Service Identifier associated with this
                mobility session from the target flows for which
                Per-Session-Agg-Max-UL/DL-Bit-Rate is measured.";
         }
     description "Per-Session-Agg-Max-Bit-Rate Value";
     }

     grouping Allocation-Retention-Priority-Value {
         leaf prioirty-level {
             type uint8 {
             range "0..15";
             }
             mandatory true;
             description
         "This is a 4-bit unsigned integer value.  It is used to decide
         whether a mobility session establishment or modification
         request can be accepted; this is typically used for
         admission control of Guaranteed Bit Rate traffic in case of
         resource limitations.  The priority level can also be used to
         decide which existing mobility session to preempt during
         resource limitations.  The priority level defines the
```

```
        relative timeliness of a resource request.

        Values 1 to 15 are defined, with value 1 as the highest level
        of priority.

        Values 1 to 8 should only be assigned for services that are
        authorized to receive prioritized treatment within an
        operator domain.  Values 9 to 15 may be assigned to resources
        that are authorized by the home network and thus applicable
        when a mobile node is roaming.";
        }
        leaf premption-capability {
            type enumeration {
            enum enabled {
          value 0;
          description "enabled";
        }
            enum disabled {
          value 1;
          description "disabled";
        }
            enum reserved1 {
          value 2;
          description "reserved1";
        }
            enum reserved2 {
          value 3;
          description "reserved2";
        }
            }
            mandatory true;
            description
       "This is a 2-bit unsigned integer value.  It defines whether a
        service data flow can get resources that were already
        assigned to another service data flow with a lower priority
        level.  The following values are defined:

         Enabled (0): This value indicates that the service data flow
         is allowed to get resources that were already assigned to
         another IP data flow with a lower priority level.

         Disabled (1): This value indicates that the service data flow
         is not allowed to get resources that were already assigned to
         another IP data flow with a lower priority level.  The values
         (2) and (3) are reserved.";
        }
        leaf premption-vulnerability {
            type enumeration {
```

```
            enum enabled {
          value 0;
          description "enabled";
        }
            enum disabled {
          value 1;
          description "disabled";
        }
            enum reserved1 {
          value 2;
          description "reserved1";
        }
            enum reserved2 {
          value 3;
          description "reserved2";
        }
            }
            mandatory true;
            description
       "This is a 2-bit unsigned integer value.  It defines whether a
         service data flow can lose the resources assigned to it in
         order to admit a service data flow with a higher priority
         level.  The following values are defined:

           Enabled (0): This value indicates that the resources
           assigned to the IP data flow can be preempted and
           allocated to a service data flow with a higher
           priority level.

           Disabled (1): This value indicates that the resources
           assigned to the IP data flow shall not be preempted and
           allocated to a service data flow with a higher priority
           level.  The values (2) and (3) are reserved.";
        }
    description "Allocation-Retention-Priority Value";
    }

    typedef Aggregate-Max-DL-Bit-Rate-Value {
        type uint32;
        description
            "This is a 32-bit unsigned integer that
         indicates the aggregate maximum downlink bit rate that is
         requested/allocated for downlink IP flows.  The measurement
         units for Aggregate-Max-DL-Bit-Rate are bits per second.";
    }

    typedef Aggregate-Max-UL-Bit-Rate-Value {
        type uint32;
        description
            "This is a 32-bit unsigned integer that
         indicates the aggregate maximum downlink bit rate that is
         requested/allocated for downlink IP flows.  The measurement
         units for Aggregate-Max-DL-Bit-Rate are bits per second.";
    }

    typedef Guaranteed-DL-Bit-Rate-Value {
        type uint32;
        description
            "This is a 32-bit unsigned integer that
         indicates the guaranteed bandwidth in bits per second for
         downlink IP flows.  The measurement units for
         Guaranteed-DL-Bit-Rate are bits per second.";
    }

    typedef Guaranteed-UL-Bit-Rate-Value {
        type uint32;
        description
            "This is a 32-bit unsigned integer that
         indicates the guaranteed bandwidth in bits per second
         for uplink IP flows.  The measurement units for
         Guaranteed-UL-Bit-Rate are bits per second.";
    }

    grouping QoS-Vendor-Specific-Attribute-Value-Base {
        leaf vendorid {
            type uint32;
            mandatory true;
            description
         "The Vendor ID is the SMI (Structure of Management
          Information) Network Management Private Enterprise Code of
          the IANA-maintained 'Private Enterprise Numbers'
          registry.";
            reference
          "'PRIVATE ENTERPRISE NUMBERS', SMI Network Management
            Private Enterprise Codes, April 2014,
             <http://www.iana.org/assignments/enterprise-numbers>";
        }
        leaf subtype {
            type uint8;
            mandatory true;
            description
          "An 8-bit field indicating the type of vendor-specific
           information carried in the option.  The namespace for this
           sub-type is managed by the vendor identified by the
           Vendor ID field.";
        }
        description
        "QoS Vendor-Specific Attribute.";
    }

    //NOTE - We do NOT add the Status Codes or other changes in
    // PMIP in this module

    //Primary Structures (groupings)
    grouping qosattribute {
        leaf attributetype {
            type identityref {
                base qos-attribute-type;
            }
            mandatory true;
            description "the attribute type";
        }

          //All of the sub-types by constraint
        choice attribute-choice {
            case per-mn-agg-max-dl-case {
                when "./attributetype = "
                   + "'Per-MN-Agg-Max-DL-Bit-Rate-type'";
                leaf per-mn-agg-max-dl {
                    type qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value;
                    description "Per-MN-Agg-Max-DL-Bit-Rate Value";
                }
                description "Per-MN-Agg-Max-DL-Bit-Rate Case";
            }
            case per-mn-agg-max-ul-case {
                when "./attributetype = "
                  + "'Per-MN-Agg-Max-UL-Bit-Rate-type'";
                leaf per-mn-agg-max-ul {
                    type qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value;
                    description "Per-MN-Agg-Max-UL-Bit-Rate Value";
                }
                description "Per-MN-Agg-Max-UL-Bit-Rate Case";
            }
            case per-session-agg-max-dl-case {
                when "./attributetype = "
                + "'Per-Session-Agg-Max-DL-Bit-Rate-type'";
                container per-session-agg-max-dl {
                    uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
                    description "Per-Session-Agg-Max-Bit-Rate Value";
                }
                description "Per-Session-Agg-Max-Bit-Rate Case";
            }
            case per-session-agg-max-ul-case {
                when "./attributetype = "
                + "'Per-Session-Agg-Max-UL-Bit-Rate-type'";
                container per-session-agg-max-ul {
                    uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
                    description "Per-Session-Agg-Max-Bit-Rate Value";
                }
                description "Per-Session-Agg-Max-Bit-Rate Case";
            }
            case allocation-retention-priority-case {
                when "./attributetype = "
                   + "'Allocation-Retention-Priority-type'";
                uses qos-pmip:Allocation-Retention-Priority-Value;
                description "Allocation-Retention-Priority Case";
            }
            case agg-max-dl-case {
                when "./attributetype = "
                  + "'Aggregate-Max-DL-Bit-Rate-type'";
                leaf agg-max-dl {
                    type qos-pmip:Aggregate-Max-DL-Bit-Rate-Value;
                    description "Aggregate-Max-DL-Bit-Rate Value";
                }
                description "Aggregate-Max-DL-Bit-Rate Case";
            }
            case agg-max-ul-case {
                when "./attributetype = "
                  + "'Aggregate-Max-UL-Bit-Rate-type'";
                leaf agg-max-ul {
                    type qos-pmip:Aggregate-Max-UL-Bit-Rate-Value;
                    description "Aggregate-Max-UL-Bit-Rate Value";
                }
                description "Aggregate-Max-UL-Bit-Rate Case";
            }
            case gbr-dl-case {
                when "./attributetype = 'Guaranteed-DL-Bit-Rate-type'";
                leaf gbr-dl {
                    type qos-pmip:Guaranteed-DL-Bit-Rate-Value;
                    description "Guaranteed-DL-Bit-Rate Value";
                }
                description "Guaranteed-DL-Bit-Rate Case";
            }
            case gbr-ul-case {
                when "./attributetype = 'Guaranteed-UL-Bit-Rate-type'";
                leaf gbr-ul {
                    type qos-pmip:Guaranteed-UL-Bit-Rate-Value;
                    description "Guaranteed-UL-Bit-Rate Value";
                }
                description "Guaranteed-UL-Bit-Rate Case";
            }
            case traffic-selector-case {
                when "./attributetype = ’QoS-Traffic-Selector-type’";
                container traffic-selector {
                    uses traffic-selectors:traffic-selector;
                    description "traffic selector";
                }
                description "traffic selector Case";
            }
            description "Attribute Value";
        }
        description "PMIP QoS Attribute";
    }

    grouping qosoption {
        leaf srid {
            type sr-id;
            mandatory true;
            description "Service Request Identifier";
        }
        leaf trafficclass {
            type traffic-class;
            mandatory true;
            description "Traffic Class";
        }
        leaf operationcode {
            type operational-code;
            mandatory true;
            description "Operation Code";
        }
        list attributes {
            unique "attributetype";
            uses qosattribute;
            min-elements 1;
            description "Attributes";
        }
        description "PMIP QoS Option";
    }
}

<CODE ENDS>
```

A.2.3. Traffic Selectors YANG Model

This module defines traffic selector types commonly used in Proxy Mobile IP (PMIP).

This module references [RFC6991].

```
 <CODE BEGINS> file "ietf-traffic-selector-types@2016-01-14.yang"
 module ietf-traffic-selector-types {
  yang-version 1;

  namespace
  "urn:ietf:params:xml:ns:yang:ietf-traffic-selector-types";

  prefix "traffic-selectors";

  import ietf-inet-types {
     prefix inet;
     revision-date 2013-07-15;
  }

  organization "IETF Distributed Mobility Management (DMM)
  Working Group";

  contact
  "WG Web: <http://tools.ietf.org/wg/netmod/>
  WG List: <mailto:netmod@ietf.org>

  WG Chair: Dapeng Liu
  <mailto:maxpassion@gmail.com>

  WG Chair: Jouni Korhonen
  <mailto:jouni.nospam@gmail.com>

  Editor: Satoru Matsushima
  <mailto:satoru.matsushima@g.softbank.co.jp>

  Editor: Lyle Bertz
  <mailto:lylebe551144@gmail.com>";

  description
  "This module contains a collection of YANG definitions for
  traffic selectors for flow bindings.

  Copyright (c) 2016 IETF Trust and the persons identified as the
  document authors. All rights reserved.

  This document is subject to BCP 78 and the IETF Trust's Legal
  Provisions Relating to IETF Documents
  (http://trustee.ietf.org/license-info) in effect on the date of
  publication of this document. Please review these documents
  carefully, as they describe your rights and restrictions with
  respect to this document. Code Components extracted from this
  document must include Simplified BSD License text as described
  in Section 4.e of the Trust Legal Provisions and are provided
  without warranty as described in the Simplified BSD License.";
  revision 2016-01-14 {
     description "Updated for IETF-PACKET-FIELDS module alignment";
     reference
         "draft-ietf-netmod-acl-model-06";
  }

  revision 2016-01-12 {
  description "Initial revision";
  reference
     "RFC 6088: Traffic Selectors for Flow Bindings";
  }

  // Identities
     identity traffic-selector-format {
         description
          "The base type for Traffic-Selector Formats";
     }

     identity ipv4-binary-selector-format {
         base traffic-selector-format;
         description
             "IPv4 Binary Traffic Selector Format";
     }

     identity ipv6-binary-selector-format {
         base traffic-selector-format;
         description
             "IPv6 Binary Traffic Selector Format";
     }

     // Type definitions and groupings
     typedef ipsec-spi {
         type uint32;
         description
             "This type defines the first 32-bit IPsec
              Security Parameter Index (SPI) value on data
              packets sent from a corresponding node to the
        mobile node as seen by the home agent. This field
        is defined in [RFC4303].";
             reference
              "RFC 4303: IP Encapsulating Security
              Payload (ESP)";
     }

     grouping traffic-selector-base {
         description "A grouping of the commen leaves between the
        v4 and v6 Traffic Selectors";
       container ipsec-spi-range {
          presence "Enables setting ipsec spi range";
          description
         "Inclusive range representing IPSec Security Parameter
         Indices to be used. When only start-spi is present, it
         represents a single spi.";
            leaf start-spi {
         type ipsec-spi;
         mandatory true;
         description
           "This field identifies the first 32-bit IPsec SPI value,
           from the range of SPI values to be matched, on data
           packets sent from a corresponding node to the mobile
           node as seen by the home agent.
           This field is defined in [RFC4303].";
              }
              leaf end-spi {
                type ipsec-spi;
                must ". >= ../start-spi" {
                  error-message
                    "The end-spi must be greater than or equal
                     to start-spi";
              }
        description
          "If more than one contiguous SPI value needs to be matched,
          then this field can be used to indicate the end value of
          a range starting from the value of the Start SPI field.
          This field MUST NOT be included unless the Start SPI
          field is included and has a value less than or equal to
          this field.

          When this field is included, the receiver will match all
          of the SPI values between fields start-spi and end-spi,
          inclusive of start-spi and end-spi.";
              }
       }
       container source-port-range {
         presence "Enables setting source port range";
         description
         "Inclusive range representing source ports to be used.
          When only start-port is present, it represents a single
                port.";
           leaf start-port {
              type inet:port-number;
              mandatory true;
              description
           "This field identifies the first 16-bit source port number,
           from the range of port numbers to be matched, on data
           packets sent from a corresponding node to the mobile node
           as seen by the home agent.
           This is from the range of port numbers defined by IANA
                 (http://www.iana.org).";
           }
           leaf end-port {
              type inet:port-number;
              must ". >= ../start-port" {
              error-message
            "The end-port must be greater than or equal to start-port";
             }
             description
           "If more than one contiguous source port number needs to be
            matched, then this field can be used to indicate the end
            value of a range starting from the value of the Start
            Port field. This field MUST NOT be included unless the
            Start Port field is included and has a value less than
            or equal to this field.

            When this field is included, the receiver will match
            all of the port numbers between fields start-port and
            end-port, inclusive of start-port and end-port.";
              }
       }
       container destination-port-range {
         presence "Enables setting destination port range";
         description
         "Inclusive range representing destination ports to be used.
         When only start-port is present, it represents a single
         port.";
               leaf start-port {
               type inet:port-number;
               mandatory true;
               description
            "This field identifies the first 16-bit destination port
            number, from the range of port numbers to be matched, on
            data packets sent from a corresponding node to the mobile
            node as seen by the home agent.";
              }
              leaf end-port {
              type inet:port-number;
              must ". >= ../start-port" {
              error-message
                "The end-port must be greater than or equal to
            start-port";
             }
             description
          "If more than one contiguous destination port number needs
          to be matched, then this field can be used to indicate
          the end value of a range starting from the value of the
          Start Destination Port field. This field MUST NOT be
          included unless the Start Port field is included and has
          a value less than or equal to this field.

          When this field is included, the receiver will match
          all of the port numbers between fields start-port and
          end-port, inclusive of start-port and end-port.";
          }
      }
   }

   grouping ipv4-binary-traffic-selector {
       container source-address-range-v4 {
         presence "Enables setting source IPv4 address range";
         description
         "Inclusive range representing IPv4 addresses to be used. When
         only start-address is present, it represents a single
         address.";
           leaf start-address {
               type inet:ipv4-address;
               mandatory true;
              description
          "This field identifies the first source address, from the range
          of 32-bit IPv4 addresses to be matched, on data packets sent
          from a corresponding node to the mobile node as seen by the
          home agent. In other words, this is one of the addresses of
          the correspondent node.";
           }
           leaf end-address {
               type inet:ipv4-address;
               description
           "If more than one contiguous source address needs to be
           matched, then this field can be used to indicate the end
           value of a range starting from the value of the Start
           Address field. This field MUST NOT be included unless the
           Start Address field is included. When this field is
           included, the receiver will match all of the addresses
           between fields start-address and end-address, inclusive of
           start-address and end-address.";
              }
       }
       container destination-address-range-v4 {
         presence "Enables setting destination IPv4 address range";
         description
           "Inclusive range representing IPv4 addresses to be used.
          When only start-address is present, it represents a
          single address.";
```

```
          leaf start-address {
              type inet:ipv4-address;
              mandatory true;
              description
          "This field identifies the first destination address, from the
          range of 32-bit IPv4 addresses to be matched, on data packets
          sent from a corresponding node to the mobile node as seen by
          the home agent. In other words, this is one of the registered
          home addresses of the mobile node.";
          }
          leaf end-address {
            type inet:ipv4-address;
              description
          "If more than one contiguous destination address needs to be
          matched, then this field can be used to indicate the end
          value of a range starting from the value of the Start
          Destination Address field. This field MUST NOT be included
          unless the Start Address field is included. When this field
          is included, the receiver will match all of the addresses
          between fields start-address and end-address, inclusive of
          start-address and end-address.";
          }
      }
      container ds-range {
        presence "Enables setting dscp range";
        description
        "Inclusive range representing DiffServ Codepoints to be used.
        When only start-ds is present, it represents a single
        Codepoint.";
        leaf start-ds {
            type inet:dscp;
            mandatory true;
            description
          "This field identifies the first differential service value,
          from the range of differential services values to be
          matched, on data packets sent from a corresponding node to
          the mobile node as seen by the home agent. Note that this
          field is called a 'Type of Service field' in [RFC0791].
          [RFC3260] then clarified that the field has been redefined
          as a 6-bit DS field with 2 bits reserved, later claimed by
          Explicit Congestion Notification (ECN) [RFC3168]. For the
          purpose of this specification, the Start DS field is 8 bits
          long, where the 6 most significant bits indicate the DS field
          to be matched and the 2 least significant bits' values MUST be
          ignored in any comparison.";
     }
     leaf end-ds {
       type inet:dscp;
       must ". >= ../start-ds" {
         error-message
           "The end-ds must be greater than or equal to start-ds";
       }
       description
        "If more than one contiguous DS value needs to be matched, then
        this field can be used to indicate the end value of a range
        starting from the value of the Start DS field. This field MUST
        NOT be included unless the Start DS field is included. When this
        field is included, it MUST be coded the same way as defined for
        start-ds. When this field is included, the receiver will match
        all of the values between fields start-ds and end-ds, inclusive
        of start-ds and end-ds.";
      }
   }
     container protocol-range {
     presence "Enables setting protocol range";
     description
       "Inclusive range representing IP protocol(s) to be used. When
        only start-protocol is present, it represents a single
        protocol.";
     leaf start-protocol {
       type uint8;
       mandatory true;
       description
         "This field identifies the first 8-bit protocol value, from the
          range of protocol values to be matched, on data packets sent
          from a corresponding node to the mobile node as seen by the
          home agent.";
         }
         leaf end-protocol {
             type uint8;
         must ". >= ../start-protocol" {
           error-message
             "The end-protocol must be greater than or equal to
           start-protocol";
         }
             description
        "If more than one contiguous protocol value needs to be matched,
        then this field can be used to indicate the end value of a range
        starting from the value of the Start Protocol field. This field
        MUST NOT be included unless the Start Protocol field is
        included. When this field is included, the receiver will match
        all of the values between fields start-protocol and
        end-protocol, inclusive of start-protocol and end-protocol.";
         }
      }
    description "ipv4 binary traffic selector";
```

```
  }

     grouping ipv6-binary-traffic-selector {
      container source-address-range-v6 {
      presence "Enables setting source IPv6 address range";
        description
        "Inclusive range representing IPv6 addresses to be used.
        When only start-address is present, it represents a
        single address.";
          leaf start-address {
              type inet:ipv6-address;
              mandatory true;
              description
         "This field identifies the first source address, from the
         range of 128-bit IPv6 addresses to be matched, on data
         packets sent from a corresponding node to the mobile node as
         seen by the home agent. In other words, this is one of the
         addresses of the correspondent node.";
       }
       leaf end-address {
         type inet:ipv6-address;
         description
          "If more than one contiguous source address needs to be
          matched, then this field can be used to indicate the end
          value of a range starting from the value of the Start
          Address field. This field MUST NOT be included unless the
          Start Address field is included. When this field is
          included, the receiver will match all of the addresses
          between fields start-address and end-address, inclusive of
          start-address and end-address .";
       }
    }
    container destination-address-range-v6 {
      presence "Enables setting destination IPv6 address range";
      description
        "Inclusive range representing IPv6 addresses to be used.
         When only start-address is present, it represents a
         single address.";
      leaf start-address {
        type inet:ipv6-address;
        mandatory true;
        description
          "This field identifies the first destination address, from
          the range of 128-bit IPv6 addresses to be matched, on data
          packets sent from a corresponding node to the mobile node as
          seen by the home agent. In other words, this is one of the
          registered home addresses of the mobile node.";
      }
      leaf end-address {
        type inet:ipv6-address;
        description
          "If more than one contiguous destination address needs to be
           matched, then this field can be used to indicate the end
           value of a range starting from the value of the Start
           Address field. This field MUST NOT be included unless the
           Start Address field is included. When this field is
           included, the receiver will match all of the addresses
           between fields start-address and end-address, inclusive of
           start-address and end-address.";
     }
   }
   container flow-label-range {
     presence "Enables setting Flow Label range";
     description
       "Inclusive range representing IPv4 addresses to be used. When
        only start-flow-label is present, it represents a single
        flow label.";
     leaf start-flow-label {
       type inet:ipv6-flow-label;
       description
         "This field identifies the first flow label value, from the
         range of flow label values to be matched, on data packets
         sent from a corresponding node to the mobile node as seen
         by the home agent. According to [RFC2460], the flow label
         is 24 bits long. For the purpose of this specification, the
         sender of this option MUST prefix the flow label value with
         8 bits of ’0’ before inserting it in the start-flow-label
         field. The receiver SHOULD ignore the first 8 bits of this
         field before using it in comparisons with flow labels in
         packets.";
     }
     leaf end-flow-label {
       type inet:ipv6-flow-label;
       must ". >= ../start-flow-label" {
         error-message
           "The end-flow-lable must be greater than or equal to
            start-flow-label";
       }
       description
         "If more than one contiguous flow label value needs to be
          matched, then this field can be used to indicate the end
          value of a range starting from the value of the Start Flow
          Label field. This field MUST NOT be included unless the
          Start Flow Label field is included. When this field is
          included, the receiver will match all of the flow label
          values between fields start-flow-label and end-flow-label,
          inclusive of start-flow-label and end-flow-label. When this
          field is included, it MUST be coded the same way as defined
          for end-flow-label.";
     }
     }
   container traffic-class-range {
     presence "Enables setting the traffic class range";
     description
      "Inclusive range representing IPv4 addresses to be used. When
       only start-traffic-class is present, it represents a single
       traffic class.";
     leaf start-traffic-class {
       type inet:dscp;
       description
        "This field identifies the first traffic class value, from the
         range of traffic class values to be matched, on data packets
         sent from a corresponding node to the mobile node as seen by
         the home agent. This field is equivalent to the Start DS field
         in the IPv4 traffic selector in Figure 1. As per RFC 3260, the
         field is defined as a 6-bit DS field with 2 bits reserved,
         later claimed by Explicit Congestion Notification (ECN)
         RFC 3168. For the purpose of this specification, the
         start-traffic-class field is 8 bits long, where the 6 most
         significant bits indicate the DS field to be matched and the 2
         least significant bits' values MUST be ignored in any
         comparison.";
       reference
        "RFC 3260: New Terminology and Clarifications for Diffserv
         RFC 3168: The Addition of Explicit Congestion Notification
         (ECN) to IP";
     }
     leaf end-traffic-class {
       type inet:dscp;
       must ". >= ../start-traffic-class" {
         error-message
           "The end-traffic-class must be greater than or equal to
            start-traffic-class";
       }
       description
        "If more than one contiguous TC value needs to be matched,
        then this field can be used to indicate the end value of a
        range starting from the value of the Start TC field. This
        field MUST NOT be included unless the Start TC field is
        included. When this field is included, it MUST be coded the
        same way as defined for start-traffic-class. When this field
        is included, the receiver will match all of the values
        between fields start-traffic-class and end-traffic-class,
        inclusive of start-traffic-class and end-traffic-class.";
     }
   }
   container next-header-range {
     presence "Enables setting Next Header range";
     description
      "Inclusive range representing Next Headers to be used. When
       only start-next-header is present, it represents a
       single Next Header.";
     leaf start-next-header {
       type uint8;
       description
        "This field identifies the first 8-bit next header value, from
        the range of next header values to be matched, on data packets
        sent from a corresponding node to the mobile node as seen by
        the home agent.";
     }
     leaf end-next-header {
       type uint8;
       must ". >= ../start-next-header" {
         error-message
           "The end-next-header must be greater than or equal to
           start-next-header";
       }
       description
       "If more than one contiguous next header value needs to be
        matched, then this field can be used to indicate the end value
        of a range starting from the value of the Start NH field. This
        field MUST NOT be included unless the Start next header field
        is included. When this field is included, the receiver will
        match all of the values between fields start-next-header and
        end-next-header, inclusive of start-next-header and
        end-next-header.";
     }
   }
   description "ipv6 binary traffic selector";
 }

     grouping traffic-selector {
         leaf ts-format {
             type identityref {
             base traffic-selector-format;
             }
             description "Traffic Selector Format";
         }
         uses traffic-selector-base {
             when "boolean(../ts-format/text() ="
             + "'ipv6-binary-selector-format') |"
             + " boolean(../ts-format/text() ="
             + " 'ipv4-binary-selector-format')";
         }
         uses ipv4-binary-traffic-selector {
             when "boolean(../ts-format/text() ="
             + " 'ipv4-binary-selector-format')";
         }
         uses ipv6-binary-traffic-selector {
             when "boolean(../ts-format/text() = "
         + "'ipv6-binary-selector-format')";
         }
     description
      "The traffic selector includes the parameters used to match
              packets for a specific flow binding.";
         reference
      "RFC 6089: Flow Bindings in Mobile IPv6 and Network
        Mobility (NEMO) Basic Support";
   }

     grouping ts-list {
         list selectors {
             key index;
             leaf index {
             type uint64;
         description "index";
             }
             uses traffic-selector;
       description "traffic selectors";
         }
     description "traffic selector list";
     }
 }
 <CODE ENDS>
```

A.2.4. FPC 3GPP Mobility YANG Model

This module defines the base protocol elements of 3GPP mobility..

This module references [RFC6991], the fpc-base, fpc-agent, ietf-traffic-selector and pmip-qos modules defined in this document.

```
   <CODE BEGINS> file "ietf-dmm-threegpp@2017-03-08.yang"
   module ietf-dmm-threegpp {
       namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-threegpp";
       prefix threegpp;

       import ietf-inet-types { prefix inet; revision-date 2013-07-15; }
       import ietf-dmm-fpc { prefix fpc; revision-date 2017-03-08; }
       import ietf-traffic-selector-types { prefix traffic-selectors;
```

```
         revision-date 2016-01-14; }
       import ietf-pmip-qos { prefix pmipqos;
           revision-date 2016-02-10; }

       organization "IETF Distributed Mobility Management (DMM)
         Working Group";

       contact
          "WG Web:   <http://tools.ietf.org/wg/netmod/>
           WG List:  <mailto:netmod@ietf.org>

           WG Chair: Dapeng Liu
                     <mailto:maxpassion@gmail.com>

           WG Chair: Jouni Korhonen
                     <mailto:jouni.nospam@gmail.com>

           Editor:   Satoru Matsushima
                     <mailto:satoru.matsushima@g.softbank.co.jp>

           Editor:   Lyle Bertz
                     <mailto:lylebe551144@gmail.com>";

       description
       "This module contains YANG definition for 3GPP Related Mobility
        Structures.

        Copyright (c) 2016 IETF Trust and the persons identified as the
        document authors. All rights reserved.

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

       revision 2017-03-08 {
           description "Version 06 updates.";
           reference "draft-ietf-dmm-fpc-cpdp-06";
       }

       revision 2016-08-03 {
           description "Initial";
           reference "draft-ietf-dmm-fpc-cpdp-04";
       }

       identity threeGPP-access-type {
         base "fpc:fpc-access-type";
         description "3GPP Access Type";
       }

       // Profile Type
       identity threeGPP-mobility {
            base "fpc:fpc-mobility-profile-type";
            description "3GPP Mobility Profile";
       }

       // Tunnel Types
       identity threeGPP-tunnel-type {
           description "3GPP Base Tunnel Type";
       }

       identity gtpv1 {
           base "threegpp:threeGPP-tunnel-type";
           description "GTP version 1 Tunnel";
       }

       identity gtpv2 {
           base "threegpp:threeGPP-tunnel-type";
           description "GTP version 2 Tunnel";
       }

       grouping teid-value {
            description "TEID value holder";
            leaf tunnel-identifier {
               type uint32;
               description "Tunnel Endpoint IDentifier (TEID)";
           }
       }

       grouping threeGPP-tunnel {
           description "3GPP Tunnel Definition";
           leaf tunnel-type {
               type identityref  {
                 base "threegpp:threeGPP-tunnel-type";
               }
               description "3GPP Tunnel Subtype";
           }
           uses threegpp:teid-value;
       }

       // QoS Profile
       identity threeGPP-qos-profile-parameters {
           base "fpc:fpc-qos-type";
           description "3GPP QoS Profile";
       }

       typedef fpc-qos-class-identifier {
           type uint8 {
               range "1..9";
           }
           description "QoS Class Identifier (QCI)";
       }

       grouping threeGPP-QoS {
           description "3GPP QoS Attributes";
           leaf qci {
               type fpc-qos-class-identifier;
               description "QCI";
           }
           leaf gbr {
               type uint32;
               description "Guaranteed Bit Rate";
           }
           leaf mbr {
               type uint32;
               description "Maximum Bit Rate";
           }
           leaf apn-ambr {
               type uint32;
               description "Access Point Name Aggregate Max Bit Rate";
           }
           leaf ue-ambr {
               type uint32;
               description "User Equipment Aggregate Max Bit Rate";
           }
           container arp {
               uses pmipqos:Allocation-Retention-Priority-Value;
               description "Allocation Retention Priority";
           }
       }

       typedef ebi-type {
         type uint8 {
           range "0..15";
         }
         description "EUTRAN Bearere Identifier (EBI) Type";
       }

      // From 3GPP TS 24.008 version 13.5.0 Release 13
      typedef component-type-enum {
           type enumeration {
               enum ipv4RemoteAddress {
                 value 16;
                 description "IPv4 Remote Address";
               }
               enum ipv4LocalAddress  {
                 value 17;
                 description "IPv4 Local Address";
               }
               enum ipv6RemoteAddress {
                 value 32;
                 description "IPv6 Remote Address";
               }
               enum ipv6RemoteAddressPrefix {
                 value 33;
                 description "IPv6 Remote Address Prefix";
               }
               enum ipv6LocalAddressPrefix {
                 value 35;
                 description "IPv6 Local Address Prefix";
               }
               enum protocolNextHeader {
                 value 48;
                 description "Protocol (IPv4) or NextHeader (IPv6)
                   value";
               }
               enum localPort {
                 value 64;
                 description "Local Port";
               }
               enum localPortRange {
                 value 65;
                 description "Local Port Range";
               }
               enum reomotePort {
                 value 80;
                 description "Remote Port";
               }
               enum remotePortRange {
                 value 81;
                 description "Remote Port Range";
               }
               enum secParamIndex {
                 value 96;
                 description "Security Parameter Index (SPI)";
               }
               enum tosTraffClass {
                 value 112;
                 description "TOS Traffic Class";
               }
               enum flowLabel {
                 value 128;
                 description "Flow Label";
               }
           }
           description "TFT Component Type";
       }

       typedef packet-filter-direction {
           type enumeration {
             enum preRel7Tft {
               value 0;
               description "Pre-Release 7 TFT";
             }
             enum uplink {
               value 1;
               description "uplink";
             }
             enum downlink {
               value 2;
               description "downlink";
             }
             enum bidirectional {
               value 3;
               description "bi-direcitonal";
             }
           }
           description "Packet Filter Direction";
       }

       typedef component-type-id {
           type uint8 {
             range "16 | 17 | 32 | 33 | 35 | 48 | 64 | 65 |"
             + " 80 | 81 | 96 | 112 | 128";
           }
           description "Specifies the Component Type";
       }

       grouping packet-filter {
         leaf direction {
             type threegpp:packet-filter-direction;
             description "Filter Direction";
         }
         leaf identifier {
             type uint8 {
               range "1..15";
             }
             description "Filter Identifier";
         }
         leaf evaluation-precedence {
             type uint8;
             description "Evaluation Precedence";
         }
         list contents {
           key component-type-identifier;
           description "Filter Contents";
           leaf component-type-identifier {
               type threegpp:component-type-id;
               description "Component Type";
           }
           choice value {
             case ipv4-local {
               leaf ipv4-local {
                 type inet:ipv4-address;
                 description "IPv4 Local Address";
               }
             }
             case ipv6-prefix-local {
               leaf ipv6-prefix-local {
                 type inet:ipv6-prefix;
                 description "IPv6 Local Prefix";
               }
             }
             case ipv4-ipv6-remote {
               leaf ipv4-ipv6-remote {
                 type inet:ip-address;
                 description "Ipv4 Ipv6 remote address";
               }
             }
             case ipv6-prefix-remote {
               leaf ipv6-prefix-remote {
                 type inet:ipv6-prefix;
                 description "IPv6 Remote Prefix";
               }
             }
             case next-header {
               leaf next-header {
                 type uint8;
                 description "Next Header";
               }
             }
             case local-port {
               leaf local-port {
                 type inet:port-number;
                 description "Local Port";
               }
             }
             case local-port-range {
               leaf local-port-lo {
                 type inet:port-number;
                 description "Local Port Min Value";
               }
               leaf local-port-hi {
                 type inet:port-number;
                 description "Local Port Max Value";
               }
             }
             case remote-port {
               leaf remote-port {
                 type inet:port-number;
                 description "Remote Port";
               }
             }
             case remote-port-range {
               leaf remote-port-lo {
                 type inet:port-number;
                 description "Remote Por Min Value";
               }
               leaf remote-port-hi {
                 type inet:port-number;
                 description "Remote Port Max Value";
               }
             }
             case ipsec-index {
               leaf ipsec-index {
                 type traffic-selectors:ipsec-spi;
                 description "IPSec Index";
               }
             }
             case traffic-class {
               leaf traffic-class {
                 type inet:dscp;
                 description "Traffic Class";
               }
             }
             case traffic-class-range {
                 leaf traffic-class-lo {
                   type inet:dscp;
                   description "Traffic Class Min Value";
                 }
                 leaf traffic-class-hi {
                   type inet:dscp;
                   description "Traffic Class Max Value";
                 }
             }
             case flow-label-type {
               leaf-list flow-label {
                 type inet:ipv6-flow-label;
                 description "Flow Label";
               }
             }
             description "Component Value";
           }
         }
         description "Packet Filter";
       }

       grouping tft {
         list packet-filters {
             key identifier;
             uses threegpp:packet-filter;
             description "List of Packet Filters";
         }
         description "Packet Filter List";
       }

       typedef imsi-type {
           type uint64;
           description
               "International Mobile Subscriber Identity (IMSI)
                 Value Type";
       }

       typedef threegpp-instr {
         type bits {
           bit assign-ip {
             position 0;
             description "Assign IP Address/Prefix";
           }
           bit assign-fteid-ip {
             position 1;
             description "Assign FTEID-IP";
           }
           bit assign-fteid-teid {
             position 2;
             description "Assign FTEID-TEID";
           }
           bit session {
             position 3;
```

```
            description "Commands apply to the Session Level";
          }
          bit uplink {
            position 4;
            description "Commands apply to the Uplink";
          }
          bit downlink {
            position 5;
            description "Commands apply to the Downlink";
          }
          bit assign-dpn {
            position 6;
            description "Assign DPN";
          }
        }
        description "Instruction Set for 3GPP R11";
      }

      // Descriptors update - goes to Entities, Configure
      //              and Configure Bundles
      augment "/fpc:tenants/fpc:tenant/fpc:fpc-policy/fpc:"
         + "descriptors/fpc:descriptor-value" {
        case threegpp-tft {
            uses threegpp:tft;
            description "3GPP TFT";
        }
        description "3GPP TFT Descriptor";
      }

      grouping threegpp-tunnel-info {
        uses threegpp:threeGPP-tunnel;
        choice tft-or-ref {
          case defined-tft {
            uses threegpp:tft;
          }
          case predefined-tft {
            leaf tft-reference {
              type fpc:fpc-identity;
              description "Pre-configured TFT";
            }
          }
          description "TFT Value";
        }
        description "3GPP TFT and Tunnel Information";
      }

      // Contexts Update - Contexts / UL / mob-profile
      augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
```

```
             + "contexts/fpc:ul/fpc:mobility-tunnel-parameters/fpc:"
             + "profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Context UL Tunnel";
         }
         augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
             + "create_or_update/fpc:contexts/fpc:ul/fpc:"
             + "mobility-tunnel-parameters/fpc:profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Create Context UL Tunnel";
         }
         augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
             + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
             + "ul/fpc:mobility-tunnel-parameters/fpc:"
             + "profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Bundles Create Context UL Tunnel";
         }
         augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
             + "create-or-update-success/fpc:contexts/fpc:"
             + "ul/fpc:mobility-tunnel-parameters/fpc:"
             + "profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Create Context UL Tunnel Response";
         }
         augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
            + "result-type/fpc:create-or-update-success/fpc:contexts/fpc:"
            + "ul/fpc:mobility-tunnel-parameters/fpc:profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Bundles Create Context UL Tunnel Response";
         }

         // Contexts Update - Contexts / DL / mob-profile
         augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
             + "contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:"
             + "profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Context DL Tunnel";
         }
         augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
             + "create_or_update/fpc:contexts/fpc:dl/fpc:"
             + "mobility-tunnel-parameters/fpc:profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Bundles Create Context DL Tunnel";
         }
         augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
             + "op_body/fpc:create_or_update/fpc:contexts/fpc:dl/fpc:"
             + "mobility-tunnel-parameters/fpc:profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Bundles Create Context DL Tunnel";
         }
         augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
             + "create-or-update-success/fpc:contexts/fpc:dl/fpc:"
             + "mobility-tunnel-parameters/fpc:profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Create Context DL Tunnel Response";
         }
         augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
             + "result-type/fpc:create-or-update-success/fpc:"
             + "contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:"
             + "profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Bundles Create Context DL Tunnel Response";
         }


         // Contexts Update -  Contexts / dpns /
         //   mobility-tunnel-parameters
         augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
             + "contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:"
             + "profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Context 3GPP TFT and Tunnel Information";
         }
         augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
             + "create_or_update/fpc:contexts/fpc:dpns/fpc:"
             + "mobility-tunnel-parameters/fpc:"
             + "profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Configure 3GPP TFT and Tunnel Information";
         }
         augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
             + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
             + "dpns/fpc:mobility-tunnel-parameters/fpc:"
             + "profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Configure Bundles 3GPP TFT and Tunnel
             Information";
         }
         augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
             + "create-or-update-success/fpc:contexts/fpc:"
             + "dpns/fpc:mobility-tunnel-parameters/fpc:"
             + "profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Configure 3GPP TFT and Tunnel Information
             Response";
         }
         augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
             + "result-type/fpc:create-or-update-success/fpc:"
             + "contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:"
             + "profile-parameters" {
           case threegpp-tunnel {
               uses threegpp:threegpp-tunnel-info;
           }
           description "Configure Bundles 3GPP TFT and Tunnel Information
             Response";
         }


         // QoS Updates - Context / UL / qosprofile
         augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
             + "contexts/fpc:ul/fpc:qos-profile-parameters/fpc:value" {
           case threegpp-qos {
               uses threegpp:threeGPP-QoS;
               description "3GPP QoS Values";
           }
           description "Context UL 3GPP QoS Values";
         }
         augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
             + "create_or_update/fpc:contexts/fpc:ul/fpc:"
             + "qos-profile-parameters/fpc:value" {
           case threegpp-qos {
               uses threegpp:threeGPP-QoS;
               description "3GPP QoS Values";
           }
           description "Configure Context UL 3GPP QoS Values";
         }
         augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
             + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
             + "ul/fpc:qos-profile-parameters/fpc:value" {
           case threegpp-qos {
               uses threegpp:threeGPP-QoS;
               description "3GPP QoS Values";
           }
           description "Configure Bundles Context UL 3GPP QoS Values";
         }
         augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
             + "create-or-update-success/fpc:contexts/fpc:ul/fpc:"
             + "qos-profile-parameters/fpc:value" {
           case threegpp-qos {
               uses threegpp:threeGPP-QoS;
               description "3GPP QoS Values";
           }
           description "Configure Context UL 3GPP QoS Values Response";
         }
         augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
             + "result-type/fpc:create-or-update-success/fpc:"
             + "contexts/fpc:ul/fpc:qos-profile-parameters/fpc:value" {
           case threegpp-qos {
               uses threegpp:threeGPP-QoS;
               description "3GPP QoS Values";
           }
           description "Configure Bundles Context UL 3GPP QoS Values
             Response";
         }

         // QoS Updates -  Context / DL / QoS Profile
         augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
             + "contexts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
           case threegpp-qos {
               uses threegpp:threeGPP-QoS;
               description "3GPP QoS Values";
           }
           description "Context DL 3GPP QoS Values";
         }
         augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
             + "create_or_update/fpc:contexts/fpc:dl/fpc:"
             + "qos-profile-parameters/fpc:value" {
           case threegpp-qos {
               uses threegpp:threeGPP-QoS;
               description "3GPP QoS Values";
           }
           description "Configure Context DL 3GPP QoS Values";
         }
         augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
             + "op_body/fpc:create_or_update/fpc:contexts/fpc:dl/fpc:"
             + "qos-profile-parameters/fpc:value" {
           case threegpp-qos {
               uses threegpp:threeGPP-QoS;
               description "3GPP QoS Values";
           }
           description "Configure Bundles Context DL 3GPP QoS Values";
         }
         augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
             + "create-or-update-success/fpc:contexts/fpc:dl/fpc:"
             + "qos-profile-parameters/fpc:value" {
           case threegpp-qos {
               uses threegpp:threeGPP-QoS;
               description "3GPP QoS Values";
           }
           description "Configure Context DL 3GPP QoS Values Response";
         }
         augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
             + "result-type/fpc:create-or-update-success/fpc:"
             + "contexts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
           case threegpp-qos {
               uses threegpp:threeGPP-QoS;
               description "3GPP QoS Values";
           }
           description "Configure Bundles Context DL 3GPP QoS Values
             Response";
         }

         grouping threegpp-properties {
           leaf imsi {
             type threegpp:imsi-type;
             description "IMSI";
           }
           leaf ebi {
             type threegpp:ebi-type;
             description "EUTRAN Bearere Identifier (EBI)";
           }
           leaf lbi {
             type threegpp:ebi-type;
             description "Linked Bearer Identifier (LBI)";
           }
           description "3GPP Mobility Session Properties";
         }

         augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts" {
           uses threegpp:threegpp-properties;
           description "3GPP Mobility Session Properties";
         }
         augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
             + "create_or_update/fpc:contexts" {
           uses threegpp:threegpp-properties;
           description "3GPP Mobility Session Properties";
         }
         augment "/fpc:configure-bundles/fpc:input/fpc:"
             + "bundles/fpc:op_body/fpc:create_or_update/fpc:contexts" {
            uses threegpp:threegpp-properties;
            description "3GPP Mobility Session Properties";
         }
         augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
             + "create-or-update-success/fpc:contexts" {
           uses threegpp:threegpp-properties;
           description "3GPP Mobility Session Properties";
         }
         augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
             + "result-type/fpc:create-or-update-success/fpc:contexts" {
            uses threegpp:threegpp-properties;
            description "3GPP Mobility Session Properties";
         }

         grouping threegpp-commandset {
           leaf instr-3gpp-mob {
             type threegpp:threegpp-instr;
             description "3GPP Specific Command Set";
           }
           description "3GPP Instructions";
         }

         augment "/fpc:configure/fpc:input/fpc:instructions/fpc:"
             + "instr-type" {
           case instr-3gpp-mob {
             uses threegpp:threegpp-commandset;
             description "3GPP Instructions";
           }
           description "Configure 3GPP Instructions";
         }
         augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
             + "create_or_update/fpc:contexts/fpc:instructions/fpc:"
             + "instr-type" {
           case instr-3gpp-mob {
             uses threegpp:threegpp-commandset;
             description "3GPP Instructions";
           }
           description "Configure 3GPP Context Instructions";
         }
         augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
             + "create-or-update-success/fpc:contexts/fpc:"
             + "instructions/fpc:instr-type" {
           case instr-3gpp-mob {
             uses threegpp:threegpp-commandset;
             description "3GPP Instructions";
           }
           description "Configure 3GPP Context Instructions Response";
         }


         augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
             + "instructions/fpc:instr-type" {
           case instr-3gpp-mob {
             uses threegpp:threegpp-commandset;
             description "3GPP Instructions";
           }
           description "Configure Bundles 3GPP Instructions";
         }
         augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
             + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
             + "instructions/fpc:instr-type" {
           case instr-3gpp-mob {
             uses threegpp:threegpp-commandset;
             description "3GPP Instructions";
           }
           description "Configure Bundles 3GPP Context Instructions";
         }
         augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
             + "result-type/fpc:create-or-update-success/fpc:"
             + "contexts/fpc:instructions/fpc:instr-type" {
           case instr-3gpp-mob {
             uses threegpp:threegpp-commandset;
             description "3GPP Instructions";
           }
           description "Configure Bundles 3GPP Context Instructions
               Response";
         }
     }
     <CODE ENDS>
```

A.2.5. FPC / PMIP Integration YANG Model

This module defines the integration between FPC and PMIP models.

This module references the fpc-base, fpc-agent, pmip-qos and traffic-selector-types module defined in this document.

```
   <CODE BEGINS> file "ietf-dmm-fpc-pmip@2017-03-08.yang"
   module ietf-dmm-fpc-pmip {
       namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-pmip";
       prefix fpc-pmip;

       import ietf-dmm-fpc { prefix fpc; revision-date 2017-03-08; }
       import ietf-pmip-qos { prefix qos-pmip; }
       import ietf-traffic-selector-types { prefix traffic-selectors; }

       organization "IETF Distributed Mobility Management (DMM)
         Working Group";

       contact
          "WG Web:   <http://tools.ietf.org/wg/netmod/>
           WG List:  <mailto:netmod@ietf.org>

           WG Chair: Dapeng Liu
                     <mailto:maxpassion@gmail.com>

           WG Chair: Jouni Korhonen
                     <mailto:jouni.nospam@gmail.com>

           Editor:   Satoru Matsushima
                     <mailto:satoru.matsushima@g.softbank.co.jp>

           Editor:   Lyle Bertz
                     <mailto:lylebe551144@gmail.com>";

       description
       "This module contains YANG definition for Forwarding Policy
        Configuration Protocol (FPCP).

        Copyright (c) 2016 IETF Trust and the persons identified as the
        document authors. All rights reserved.

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

       revision 2017-03-08 {
           description "Version 06 update. Adds predfined selector.";
           reference "draft-ietf-dmm-fpc-cpdp-06";
       }

       revision 2016-01-19 {
           description "Changes based on -01 version of FPCP draft.";
           reference "draft-ietf-dmm-fpc-cpdp-01";
       }

       identity ietf-pmip-access-type {
         base "fpc:fpc-access-type";
         description "PMIP Access";
       }

       identity fpcp-qos-index-pmip {
           base "fpc:fpc-qos-type";
           description "PMIP QoS";
       }
       identity traffic-selector-mip6 {
           base "fpc:fpc-descriptor-type";
           description "MIP6 Traffic Selector";
       }
       identity ietf-pmip {
           base "fpc:fpc-mobility-profile-type";
           description "PMIP Mobility";
       }

       identity pmip-tunnel-type {
           description "PMIP Tunnel Type";
       }
       identity grev1 {
           base "fpc-pmip:pmip-tunnel-type";
           description "GRE v1";
       }
       identity grev2 {
           base "fpc-pmip:pmip-tunnel-type";
           description "GRE v2";
       }
       identity ipinip {
           base "fpc-pmip:pmip-tunnel-type";
           description "IP in IP";
       }
       grouping pmip-mobility {
           leaf type {
               type identityref {
                   base "fpc-pmip:pmip-tunnel-type";
               }
               description "PMIP Mobility";
           }
           choice value {
               case gre {
                   leaf key {
                       type uint32;
                       description "GRE_KEY";
                   }
                   description "GRE Value";
               }
               description "PMIP Mobility value";
           }
           description "PMIP Mobility Value";
       }

       typedef pmip-instr {
         type bits {
           bit assign-ip {
             position 0;
             description "Assign IP";
           }
           bit assign-dpn {
             position 1;
             description "Assign DPN";
           }
           bit session {
             position 2;
             description "Session Level";
           }
           bit uplink {
             position 3;
             description "Uplink";
           }
           bit downlink {
             position 4;
             description "Downlink";
           }
         }
         description "Instruction Set for PMIP";
       }

       // Descriptors update - goes to Entities, Configure and
       // Configure Bundles
       augment "/fpc:tenants/fpc:tenant/fpc:fpc-policy/"
         + "fpc:descriptors/fpc:descriptor-value" {
         case pmip-selector {
             uses traffic-selectors:traffic-selector;
             description "PMIP Selector";
         }
         description "Policy Descriptor";
       }

       grouping pmip-tunnel-info {
           uses fpc-pmip:pmip-mobility;
           choice pmiptunnel-or-ref {
             case defined-selector {
               uses traffic-selectors:traffic-selector;
             }
             case predefined-selector {
               leaf selector-reference {
                 type fpc:fpc-identity;
                 description "Pre-configured selector";
               }
             }
             description "Traffic Selector Value";
           }
           description "PMIP Tunnel Information";
       }

       // Contexts Update - Contexts/UL/mob-profile, Contexts/DL/
       //   mob-profile and Contexts/dpns/mobility-tunnel-parameters
       augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
           + "contexts/fpc:ul/fpc:mobility-tunnel-parameters/fpc:"
           + "profile-parameters" {
         case pmip-tunnel {
           uses fpc-pmip:pmip-tunnel-info;
         }
         description "Context UL Mobility";
       }
       augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
           + "create_or_update/fpc:contexts/fpc:ul/fpc:"
           + "mobility-tunnel-parameters/fpc:"
           + "profile-parameters" {
         case pmip-tunnel {
           uses fpc-pmip:pmip-tunnel-info;
         }
         description "CONF Context UL Mobility";
       }
       augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
           + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
           + "ul/fpc:mobility-tunnel-parameters/fpc:"
           + "profile-parameters" {
         case pmip-tunnel {
           uses fpc-pmip:pmip-tunnel-info;
         }
         description "CONF_BUNDLES Context UL Mobility";
       }

       augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
           + "contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:"
           + "profile-parameters" {
         case pmip-tunnel {
           uses fpc-pmip:pmip-tunnel-info;
         }
         description "Context DL Mobility";
       }
       augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
           + "create_or_update/fpc:contexts/fpc:dl/fpc:"
           + "mobility-tunnel-parameters/fpc:"
           + "profile-parameters" {
         case pmip-tunnel {
           uses fpc-pmip:pmip-tunnel-info;
         }
         description "CONF Context DL Mobility";
       }
       augment "/fpc:configure-bundles/fpc:input/fpc:"
           + "bundles/fpc:op_body/fpc:create_or_update/fpc:"
           + "contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:"
           + "profile-parameters" {
         case pmip-tunnel {
           uses fpc-pmip:pmip-tunnel-info;
         }
         description "CONF_BUNDLES Context DL Mobility";
       }

       augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
           + "contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:"
           + "profile-parameters" {
         case pmip-tunnel {
           uses fpc-pmip:pmip-tunnel-info;
         }
         description "Context DPN Mobility";
       }
       augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
           + "create_or_update/fpc:contexts/fpc:dpns/fpc:"
           + "mobility-tunnel-parameters/fpc:profile-parameters" {
         case pmip-tunnel {
           uses fpc-pmip:pmip-tunnel-info;
         }
         description "CONF Context DPN Mobility";
       }
       augment "/fpc:configure-bundles/fpc:input/fpc:"
           + "bundles/fpc:op_body/fpc:create_or_update/fpc:"
           + "contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:"
           + "profile-parameters" {
         case pmip-tunnel {
           uses fpc-pmip:pmip-tunnel-info;
         }
         description "CONF_BUNDLES Context DPN Mobility";
       }

       // QoS Updates - Context / UL / qosprofile, Context / DL /
       // QoS Profile
       augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
           + "contexts/fpc:ul/fpc:qos-profile-parameters/fpc:value" {
         case qos-pmip {
             uses qos-pmip:qosattribute;
             description "PMIP QoS Information";
         }
         description "Context UL QoS";
       }
       augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
           + "create_or_update/fpc:contexts/fpc:ul/fpc:"
           + "qos-profile-parameters/fpc:value" {
         case qos-pmip {
             uses qos-pmip:qosattribute;
             description "PMIP QoS Information";
         }
         description "CONF Context UL QoS";
       }
       augment "/fpc:configure-bundles/fpc:input/fpc:"
           + "bundles/fpc:op_body/fpc:create_or_update/fpc:"
           + "contexts/fpc:ul/fpc:qos-profile-parameters/fpc:value" {
         case qos-pmip {
             uses qos-pmip:qosattribute;
             description "PMIP QoS Information";
         }
         description "CONF_BUNDLES Context UL QoS";
       }

       augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
           + "contexts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
         case qos-pmip {
             uses qos-pmip:qosattribute;
             description "PMIP QoS Information";
         }
         description "Context DL QoS";
       }
       augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
           + "create_or_update/fpc:contexts/fpc:dl/fpc:"
           + "qos-profile-parameters/fpc:value" {
         case qos-pmip {
             uses qos-pmip:qosattribute;
             description "PMIP QoS Information";
         }
         description "CONF Context DL QoS";
       }
       augment "/fpc:configure-bundles/fpc:input/fpc:"
           + "bundles/fpc:op_body/fpc:create_or_update/fpc:"
           + "contexts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
         case qos-pmip {
             uses qos-pmip:qosattribute;
             description "PMIP QoS Information";
         }
         description "CONF_BUNDLES Context DL QoS";
       }

       grouping pmip-commandset {
         leaf instr-pmip {
           type fpc-pmip:pmip-instr;
           description "PMIP Instructions";
         }
         description "PMIP Commandset";
       }

       // Instructions Update - OP BODY, Context, Port
       augment "/fpc:configure/fpc:input/fpc:instructions/fpc:"
           + "instr-type" {
         case pmip-instr {
           uses fpc-pmip:pmip-commandset;
           description "PMIP Commandset";
         }
         description "CONF Instructions";
       }
       augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
           + "create_or_update/fpc:contexts/fpc:instructions/fpc:"
           + "instr-type" {
         case pmip-instr {
           uses fpc-pmip:pmip-commandset;
           description "PMIP Commandset";
         }
         description "CONF Context Instructions";
       }
       augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
```

```
            + "create-or-update-success/fpc:contexts/fpc:"
            + "instructions/fpc:instr-type" {
          case pmip-instr {
            uses fpc-pmip:pmip-commandset;
            description "PMIP Commandset";
          }
          description "CONF Result Context Instructions";
        }

        augment "/fpc:configure-bundles/fpc:input/fpc:"
            + "bundles/fpc:instructions/fpc:instr-type" {
          case pmip-instr {
            uses fpc-pmip:pmip-commandset;
            description "PMIP Commandset";
          }
          description "CONF_BUNDLES Instructions";
        }
        augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
            + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
            + "instructions/fpc:instr-type" {
          case pmip-instr {
            uses fpc-pmip:pmip-commandset;
            description "PMIP Commandset";
          }
          description "CONF_BUNDLES Context Instructions";
        }
        augment "/fpc:configure-bundles/fpc:output/fpc:"
            + "bundles/fpc:result-type/fpc:create-or-update-success/fpc:"
            + "contexts/fpc:instructions/fpc:instr-type" {
          case pmip-instr {
            uses fpc-pmip:pmip-commandset;
            description "PMIP Commandset";
          }
          description "CONF_BUNDLES Result Context Instructions";
        }
    }
    <CODE ENDS>
```

A.2.6. FPC Policy Extension YANG Model

This module defines extensions to FPC policy structures.

This module references [RFC6991], the fpc-base and fpcagent module defined in this document.

```
    <CODE BEGINS> file "ietf-dmm-fpc-policyext@2017-03-08.yang"
    module ietf-dmm-fpc-policyext {
        namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-policyext";
```

```
       prefix fpcpolicyext;

       import ietf-dmm-fpc { prefix fpc; revision-date 2017-03-08; }
       import ietf-inet-types { prefix inet; revision-date 2013-07-15; }

       organization "IETF Distributed Mobility Management (DMM)
         Working Group";

       contact
          "WG Web:   <http://tools.ietf.org/wg/netmod/>
           WG List:  <mailto:netmod@ietf.org>

           WG Chair: Dapeng Liu
                     <mailto:maxpassion@gmail.com>

           WG Chair: Jouni Korhonen
                     <mailto:jouni.nospam@gmail.com>

           Editor:   Satoru Matsushima
                     <mailto:satoru.matsushima@g.softbank.co.jp>

           Editor:   Lyle Bertz
                     <mailto:lylebe551144@gmail.com>";

       description
       "This module contains YANG definition for Forwarding Policy
        Configuration Protocol (FPCP) common Policy Action and
        Descriptor extensions.

        Copyright (c) 2016 IETF Trust and the persons identified as the
        document authors. All rights reserved.

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

       revision 2017-03-08 {
           description "Version 06 update.";
           reference "draft-ietf-dmm-fpc-cpdp-06";
       }

       revision 2016-08-03 {
           description "Changes based on -04 version of FPC draft.";
           reference "draft-ietf-dmm-fpc-cpdp-04";
       }

       identity service-function {
           base "fpc:fpc-descriptor-type";
           description "Base Identifier for Service Functions.";
       }
       identity napt-service {
           base "service-function";
           description "NAPT Service";
       }
       grouping simple-nat {
         leaf outbound-nat-address {
           type inet:ip-address;
           description "Outbound NAT Address";
         }
         description "Simple NAT value";
       }

       identity nat-service {
           base "service-function";
           description "NAT Service";
       }
       grouping simple-napt {
         leaf source-port {
           type inet:port-number;
           description "Source Port";
         }
         leaf outbound-napt-address {
           type inet:ip-address;
           description "Outbound NAPT Address";
         }
         leaf destination-port {
           type inet:port-number;
           description "Destination Port";
         }
         description "Simple NAPT Configuration";
       }

       identity copy-forward {
         base "fpc:fpc-descriptor-type";
         description "Copies a packet then forwards to a specific
           destination";
       }
       grouping copy-forward {
         container destination {
           choice value {
             case port-ref {
               leaf port-ref {
                 type fpc:fpc-vport-id;
                 description "Port";
               }
               description "Port Forward Case";
             }
             case context-ref {
               leaf context-ref {
                 type fpc:fpc-context-id;
                 description "Context";
               }
               description "Context Forward Case";
             }
             description "Copy Forward Value";
           }
           description "destination";
         }
         description "Copy Then Forward to Port/Context Action";
       }

       augment "/fpc:tenants/fpc:tenant/fpc:fpc-policy/fpc:actions/fpc:"
           + "action-value" {
         case simple-nat {
             uses fpcpolicyext:simple-nat;
             description "Simple NAT value";
         }
         case simple-napt {
             uses fpcpolicyext:simple-napt;
             description "Simple NAPT Value";
         }
         case copy-forward {
             uses fpcpolicyext:copy-forward;
             description "Copy Forward Value";
         }
         description "Policy Actions Augmentations";
       }

       grouping prefix-traffic-descriptor {
           leaf destination-ip {
               type inet:ip-prefix;
               description "Rule of destination IP";
           }
           leaf source-ip {
               type inet:ip-prefix;
               description "Rule of source IP";
           }
           description
           "Traffic descriptor group collects parameters to
            identify target traffic flow.  It represents
            source/destination as IP prefixes";
       }

       augment "/fpc:tenants/fpc:tenant/fpc:fpc-policy/fpc:"
           + "descriptors/fpc:descriptor-value" {
         case prefix-descriptor {
             uses fpcpolicyext:prefix-traffic-descriptor;
             description "traffic descriptor value";
         }
         description "Descriptor Augments";
       }
   }
   <CODE ENDS>
```

A.3. FPC YANG Data Model Structure

This section only shows the structure for FPC YANG model.

```
   module: ietf-dmm-fpc
   +--rw tenants
   |  +--rw tenant* [tenant-id]
   |     +--rw tenant-id       fpc:fpc-identity
   |     +--rw fpc-policy
   |     |  +--rw policy-groups* [policy-group-id]
   |     |  |  +--rw policy-group-id    fpc:fpc-policy-group-id
   |     |  |  +--rw policies*          fpc:fpc-policy-id
   |     |  +--rw policies* [policy-id]
   |     |  |  +--rw policy-id    fpc:fpc-policy-id
   |     |  |  +--rw rules* [order]
   |     |  |     +--rw order          uint32
   |     |  |     +--rw descriptors* [descriptor-id]
   |     |  |     |  +--rw descriptor-id    fpc:fpc-identity
   |     |  |     |  +--rw direction?       fpc:fpc-direction
   |     |  |     +--rw actions* [action-id]
   |     |  |        +--rw action-order?   uint32
   |     |  |        +--rw action-id       fpc:fpc-action-id-type
   |     |  +--rw descriptors* [descriptor-id]
   |     |  |  +--rw descriptor-id      fpc:fpc-identity
   |     |  |  +--rw descriptor-type    identityref
   |     |  |  +--rw (descriptor-value)?
   |     |  |     +--:(all-traffic)
   |     |  |        +--rw all-traffic?       empty
   |     |  +--rw actions* [action-id]
   |     |     +--rw action-id      fpc:fpc-action-id-type
   |     |     +--rw action-type    identityref
   |     |     +--rw (action-value)?
   |     |        +--:(drop)
   |     |           +--rw drop?          empty
   |     +--ro fpc-mobility
   |     |  +--ro contexts* [context-id]
   |     |  |  +--ro context-id              fpc:fpc-context-id
   |     |  |  +--ro vports*                 fpc:fpc-vport-id
   |     |  |  +--ro dpn-group?              fpc:fpc-dpn-group-id
   |     |  |  +--ro delegated-ip-prefixes*   inet:ip-prefix
   |     |  |  +--ro ul {fpc:fpc-basic-agent}?
   |     |  |  |  +--ro tunnel-local-address?         inet:ip-address
   |     |  |  |  +--ro tunnel-remote-address?        inet:ip-address
   |     |  |  |  +--ro mtu-size?                     uint32
   |     |  |  |  +--ro mobility-tunnel-parameters
   |     |  |  |  |  +--ro (profile-parameters)?
   |     |  |  |  |     +--:(nothing)
   |     |  |  |  |        +--ro none?   empty
   |     |  |  |  +--ro nexthop
   |     |  |  |  |  +--ro nexthop-type?   identityref
   |     |  |  |  |  +--ro (nexthop-value)?
   |     |  |  |  |     +--:(ip-nexthop)
   |     |  |  |  |     |  +--ro ip?             inet:ip-address
   |     |  |  |  |     +--:(macaddress-nexthop)
   |     |  |  |  |     |  +--ro macaddress?     ytypes:mac-address
   |     |  |  |  |     +--:(servicepath-nexthop)
   |     |  |  |  |     |  +--ro servicepath?  fpc:fpc-service-path-id
   |     |  |  |  |     +--:(mplslabel-nexthop)
   |     |  |  |  |     |  +--ro lsp?            fpc:fpc-mpls-label
   |     |  |  |  |     +--:(if-nexthop)
   |     |  |  |  |        +--ro if-index?       uint16
   |     |  |  |  +--ro qos-profile-parameters
   |     |  |  |  |  +--ro qos-type?   identityref
   |     |  |  |  |  +--ro (value)?
   |     |  |  |  +--ro dpn-parameters
   |     |  |  |  +--ro vendor-parameters* [vendor-id vendor-type]
   |     |  |  |     +--ro vendor-id      fpc:fpc-identity
   |     |  |  |     +--ro vendor-type    identityref
   |     |  |  |     +--ro (value)?
   |     |  |  |        +--:(empty-type)
   |     |  |  |           +--ro empty-type?    empty
   |     |  |  +--ro dl {fpc:fpc-basic-agent}?
   |     |  |  |  +--ro tunnel-local-address?         inet:ip-address
   |     |  |  |  +--ro tunnel-remote-address?        inet:ip-address
   |     |  |  |  +--ro mtu-size?                     uint32
   |     |  |  |  +--ro mobility-tunnel-parameters
   |     |  |  |  |  +--ro (profile-parameters)?
   |     |  |  |  |     +--:(nothing)
   |     |  |  |  |        +--ro none?   empty
   |     |  |  |  +--ro nexthop
   |     |  |  |  |  +--ro nexthop-type?   identityref
   |     |  |  |  |  +--ro (nexthop-value)?
   |     |  |  |  |     +--:(ip-nexthop)
   |     |  |  |  |     |  +--ro ip?             inet:ip-address
   |     |  |  |  |     +--:(macaddress-nexthop)
   |     |  |  |  |     |  +--ro macaddress?     ytypes:mac-address
   |     |  |  |  |     +--:(servicepath-nexthop)
   |     |  |  |  |     |  +--ro servicepath?  fpc:fpc-service-path-id
   |     |  |  |  |     +--:(mplslabel-nexthop)
   |     |  |  |  |     |  +--ro lsp?            fpc:fpc-mpls-label
   |     |  |  |  |     +--:(if-nexthop)
   |     |  |  |  |        +--ro if-index?       uint16
   |     |  |  |  +--ro qos-profile-parameters
   |     |  |  |  |  +--ro qos-type?   identityref
   |     |  |  |  |  +--ro (value)?
   |     |  |  |  +--ro dpn-parameters
   |     |  |  |  +--ro vendor-parameters* [vendor-id vendor-type]
   |     |  |  |     +--ro vendor-id      fpc:fpc-identity
   |     |  |  |     +--ro vendor-type    identityref
   |     |  |  |     +--ro (value)?
   |     |  |  |        +--:(empty-type)
   |     |  |  |           +--ro empty-type?    empty
   |     |  |  +--ro dpns* [dpn-id direction] {fpc:fpc-multi-dpn}?
   |     |  |  |  +--ro dpn-id                        fpc:fpc-dpn-id
   |     |  |  |  +--ro direction                     fpc:fpc-direction
   |     |  |  |  +--ro tunnel-local-address?         inet:ip-address
   |     |  |  |  +--ro tunnel-remote-address?        inet:ip-address
   |     |  |  |  +--ro mtu-size?                     uint32
   |     |  |  |  +--ro mobility-tunnel-parameters
   |     |  |  |  |  +--ro (profile-parameters)?
   |     |  |  |  |     +--:(nothing)
   |     |  |  |  |        +--ro none?   empty
   |     |  |  |  +--ro nexthop
   |     |  |  |  |  +--ro nexthop-type?   identityref
   |     |  |  |  |  +--ro (nexthop-value)?
   |     |  |  |  |     +--:(ip-nexthop)
   |     |  |  |  |     |  +--ro ip?             inet:ip-address
   |     |  |  |  |     +--:(macaddress-nexthop)
   |     |  |  |  |     |  +--ro macaddress?     ytypes:mac-address
   |     |  |  |  |     +--:(servicepath-nexthop)
   |     |  |  |  |     |  +--ro servicepath? fpc:fpc-service-path-id
   |     |  |  |  |     +--:(mplslabel-nexthop)
   |     |  |  |  |     |  +--ro lsp?            fpc:fpc-mpls-label
   |     |  |  |  |     +--:(if-nexthop)
   |     |  |  |  |        +--ro if-index?       uint16
   |     |  |  |  +--ro qos-profile-parameters
   |     |  |  |  |  +--ro qos-type?   identityref
   |     |  |  |  |  +--ro (value)?
   |     |  |  |  +--ro dpn-parameters
   |     |  |  |  +--ro vendor-parameters* [vendor-id vendor-type]
   |     |  |  |     +--ro vendor-id      fpc:fpc-identity
   |     |  |  |     +--ro vendor-type    identityref
   |     |  |  |     +--ro (value)?
   |     |  |  |        +--:(empty-type)
   |     |  |  |           +--ro empty-type?    empty
   |     |  |  +--ro parent-context?          fpc:fpc-context-id
   |     |  +--ro vports* [vport-id]
   |     |  |  +--ro vport-id         fpc:fpc-vport-id
   |     |  |  +--ro policy-groups*   fpc:fpc-policy-group-id
   |     |  +--ro monitors*
   |     |     +--ro monitor-id?         fpc:fpc-identity
   |     |     +--ro target?             fpc-identity
   |     |     +--ro (event-config-value)?
   |     |        +--:(periodic-config)
   |     |        |  +--ro period?             uint32
   |     |        +--:(threshold-config)
   |     |        |  +--ro lo-thresh?          uint32
   |     |        |  +--ro hi-thresh?          uint32
   |     |        +--:(scheduled-config)
   |     |        |  +--ro report-time?        uint32
   |     |        +--:(events-config-ident)
   |     |        |  +--ro event-identities*   identityref
   |     |        +--:(events-config)
   |     |           +--ro event-ids*          uint32
   |     +--rw fpc-topology
   |        +--rw domains* [domain-id]
   |        |  +--rw domain-id           fpc:fpc-domain-id
   |        |  +--rw domain-name?        string
   |        |  +--rw domain-type?        string
   |        |  +--rw domain-reference?   instance-identifier
   |        |  +--rw basename?           fpc:fpc-identity
   |        |  |       {fpc:fpc-basename-registry}?
   |        |  +--rw base-state?         string
   |        |  |       {fpc:fpc-basename-registry}?
   |        |  +--rw base-checkpoint?    string
   |        |          {fpc:fpc-basename-registry}?
   |        +--rw dpn-id?              fpc:fpc-dpn-id
   |        |       {fpc:fpc-basic-agent}?
   |        +--rw control-protocols*   identityref
   |        |       {fpc:fpc-basic-agent}?
   |        +--rw dpn-groups* [dpn-group-id] {fpc:fpc-multi-dpn}?
   |        |  +--rw dpn-group-id        fpc:fpc-dpn-group-id
   |        |  +--rw data-plane-role?    identityref
   |        |  +--rw access-type?        identityref
   |        |  +--rw mobility-profile?   identityref
   |        |  +--rw dpn-group-peers* [remote-dpn-group-id]
   |        |  |  +--rw remote-dpn-group-id     fpc:fpc-dpn-group-id
   |        |  |  +--rw remote-mobility-profile?   identityref
   |        |  |  +--rw remote-data-plane-role?    identityref
   |        |  |  +--rw remote-endpoint-address?   inet:ip-address
   |        |  |  +--rw local-endpoint-address?    inet:ip-address
   |        |  |  +--rw mtu-size?                  uint32
   |        |  +--rw domains* [domain-id]
   |        |     +--rw domain-id           fpc:fpc-domain-id
   |        |     +--rw domain-name?        string
   |        |     +--rw domain-type?        string
   |        |     +--rw domain-reference?   instance-identifier
   |        |     +--rw basename?           fpc:fpc-identity
   |        |     |       {fpc:fpc-basename-registry}?
   |        |     +--rw base-state?         string
   |        |     |       {fpc:fpc-basename-registry}?
   |        |     +--rw base-checkpoint?    string
   |        |             {fpc:fpc-basename-registry}?
   |        +--rw dpns* [dpn-id] {fpc:fpc-multi-dpn}?
   |           +--rw dpn-id            fpc:fpc-dpn-id
   |           +--rw dpn-name?         string
   |           +--rw dpn-groups*       fpc:fpc-dpn-group-id
   |           +--rw node-reference?   instance-identifier
   +--rw fpc-agent-info
      +--rw supported-features*      string
      +--rw supported-events* [event]
      |  +--rw event       identityref
      |  +--rw event-id?   fpc:event-type-id
      +--rw supported-error-types* [error-type]
         +--rw error-type       identityref
         +--rw error-type-id?   fpc:error-type-id
```

Figure 28: YANG FPC Agent Tree

Authors' Addresses

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

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

Lyle Bertz
6220 Sprint Parkway
Overland Park KS, 66251
USA

Email: lylebe551144@gmail.com

Marco Liebsch
NEC Laboratories Europe
NEC Europe Ltd.
Kurfuersten-Anlage 36
D-69115 Heidelberg
Germany

Phone: +49 6221 4342146
Email: liebsch@neclab.eu

Sri Gundavelli
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sgundave@cisco.com

Danny Moses

Email: danny.moses@intel.com

Charles E. Perkins
Futurewei Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1-408-330-4586
Email: charliep@computer.org

DMM Working Group   S. Matsushima
Internet-Draft   SoftBank
Intended status: Standards Track   L. Bertz
Expires: December 20, 2018   Sprint
                                                              M. Liebsch
                                                                     NEC
                                                           S. Gundavelli
                                                                   Cisco
                                                                D. Moses
                                                       Intel Corporation
                                                              C. Perkins
                                                               Futurewei
                                                           June 18, 2018

Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-12

Abstract

This document describes a way, called Forwarding Policy Configuration (FPC) to manage the separation of data-plane and control-plane. FPC defines a flexible mobility management system using FPC agent and FPC client functions. A FPC agent provides an abstract interface to the data-plane. The FPC client configures data-plane nodes by using the functions and abstractions provided by the FPC agent for the data-plane nodes. The data-plane abstractions presented in this document are extensible in order to support many different types of mobility management systems and data-plane functions.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes Forwarding Policy Configuration (FPC), a system for managing the separation of control-plane and data-plane. FPC enables flexible mobility management using FPC client and FPC agent functions. A FPC agent exports an abstract interface representing the data-plane. To configure data-plane nodes and functions, the FPC client uses the interface to the data-plane offered by the FPC agent.

Control planes of mobility management systems, or related applications which require data-plane control, can utilize the FPC client at various levels of abstraction. FPC operations are capable of directly configuring a single Data-Plane Node (DPN), as well as multiple DPNs, as determined by the data-plane models exported by the FPC agent.

A FPC agent represents the data-plane operation according to several basic information models. A FPC agent also provides access to Monitors, which produce reports when triggered by events or FPC Client requests regarding Mobility Contexts, DPNs or the Agent.

To manage mobility sessions, the FPC client assembles applicable sets of forwarding policies from the data model, and configures them on the appropriate FPC Agent. The Agent then renders those policies into specific configurations for each DPN at which mobile nodes are attached. The specific protocols and configurations to configure a DPN from a FPC Agent are outside the scope of this document.

A DPN is a logical entity that performs data-plane operations (packet movement and management). It may represent a physical DPN unit, a sub-function of a physical DPN or a collection of physical DPNs (i.e., a "virtual DPN"). A DPN may be virtual -- it may export the FPC DPN Agent interface, but be implemented as software that controls other data-plane hardware or modules that may or may not be FPC-compliant. In this document, DPNs are specified without regard for whether the implementation is virtual or physical. DPNs are connected to provide mobility management systems such as access networks, anchors and domains. The FPC agent interface enables establishment of a topology for the forwarding plane.

When a DPN is mapped to physical data-plane equipment, the FPC client can have complete knowledge of the DPN architecture, and use that information to perform DPN selection for specific sessions. On the other hand, when a virtual DPN is mapped to a collection of physical DPNs, the FPC client cannot select a specific physical DPN because it is hidden by the abstraction; only the FPC Agent can address the specific associated physical DPNs. Network architects have the flexibility to determine which DPN-selection capabilities are performed by the FPC Agent (distributed) and which by the FPC client (centralized). In this way, overlay networks can be configured without disclosing detailed knowledge of the underlying hardware to the FPC client and applications.

The abstractions in this document are designed to support many different mobility management systems and data-plane functions. The architecture and protocol design of FPC is not tied to specific types of access technologies and mobility protocols.

2. 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].

Attribute Expression:	The definition of a template Property. This includes setting the type, current value, default value and if the attribute is static, i.e. can no longer be changed.

Domain:	One or more DPNs that form a logical partition of network resources (e.g., a data-plane network under common network administration). A FPC client (e.g., a mobility management system) may utilize a single or multiple domains.
DPN:	A data-plane node (DPN) is capable of performing data-plane features. For example, DPNs may be switches or routers, regardless of whether they are realized as hardware or purely in software.
FPC Client:	A FPC Client is integrated with a mobility management system or related application, enabling control over forwarding policy, mobility sessions and DPNs via a FPC Agent.
Mobility Context:	A Mobility Context contains the data-plane information necessary to efficiently send and receive traffic from a mobile node. This includes policies that are created or modified during the network’s operation - in most cases, on a per-flow or per session basis. A Mobility-Context represents the mobility sessions (or flows) which are active on a mobile node. This includes associated runtime attributes, such as tunnel endpoints, tunnel identifiers, delegated prefix(es), routing information, etc. Mobility-Contexts are associated to specific DPNs. Some pre-defined Policies may apply during mobility signaling requests. The Mobility Context supplies information about the policy settings specific to a mobile node and its flows; this information is often quite dynamic.
Mobility Session:	Traffic to/from a mobile node that is expected to survive reconnection events.
Monitor:	A reporting mechanism for a list of events that trigger notification messages from a FPC Agent to a FPC Client.
Policy:	A Policy determines the mechanisms for managing specific traffic flows or packets. Policies specify QoS, rewriting rules for

	packet processing, etc. A Policy consists of one or more rules. Each rule is composed of a Descriptor and Actions. The Descriptor in a rule identifies packets (e.g., traffic flows), and the Actions apply treatments to packets that match the Descriptor in the rule. Policies can apply to Domains, DPNs, Mobile Nodes, Service-Groups, or particular Flows on a Mobile Node.
Property:	An attribute-value pair for an instance of a FPC entity.
Service-Group:	A set of DPN interfaces that support a specific data-plane purpose, e.g. inbound/ outbound, roaming, subnetwork with common specific configuration, etc.
Template:	A recipe for instantiating FPC entities. Template definitions are accessible (by name or by a key) in an indexed set. A Template is used to create specific instances (e.g., specific policies) by assigning appropriate values into the Template definition via Attribute Expression.
Template Configuration	The process by which a Template is referenced (by name or by key) and Attribute Expressions are created that change the value, default value or static nature of the Attribute, if permitted. If the Template is Extensible, new attributes MAY be added.
Tenant:	An operational entity that manages mobility management systems or applications which require data-plane functions. A Tenant defines a global namespace for all entities owned by the Tenant enabling its entities to be used by multiple FPC Clients across multiple FPC Agents.
Topology:	The DPNs and the links between them. For example, access nodes may be assigned to a Service-Group which peers to a Service-Group of anchor nodes.

3. FPC Design Objectives and Deployment

Using FPC, mobility control-planes and applications can configure DPNs to perform various mobility management roles as described in [I-D.ietf-dmm-deployment-models]. This fulfills the requirements described in [RFC7333].

This document defines FPC Agent and FPC Client, as well as the information models that they use. The attributes defining those models serve as the protocol elements for the interface between the FPC Agent and the FPC Client.

Mobility control-plane applications integrate features offered by the FPC Client. The FPC Client connects to FPC Agent functions. The Client and the Agent communicate based on information models described in Section 4. The models allow the control-plane to configure forwarding policies on the Agent for data-plane communications with mobile nodes.

Once the Topology of DPN(s) and domains are defined on an Agent for a data plane, the DPNs in the topology are available for further configuration. The FPC Agent connects those DPNs to manage their configurations.

A FPC Agent configures and manages its DPN(s) according to forwarding policies requested and Attributes provided by the FPC Client. Configuration commands used by the FPC agent to configure its DPN node(s) may be specific to the DPN implementation; consequently the method by which the FPC Agent carries out the specific configuration for its DPN(s) is out of scope for this document. Along with the data models, the FPC Client (on behalf of control-plane and applications) requests that the Agent configures Policies prior to the time when the DPNs start forwarding data for their mobility sessions.

This architecture is illustrated in Figure 1. A FPC Agent may be implemented in a network controller that handles multiple DPNs, or (more simply) an FPC Agent may itself be integrated into a DPN.

This document does not specify a protocol for the FPC interface; it is out of scope. However, an implementation must support the FPC transactions described in Section 5.

```
                           +-------------------------+
                           | Mobility Control-Plane  |
                           |          and            |
                           |      Applications       |
                           |+-----------------------+|
                           ||      FPC Client       ||
                           |+----------^------------+|
                           +-----------|-------------+
               FPC interface protocol  |
                       +---------------+-----------------+
                       |                                 |
         Network       |                                 |
         Controller    |                      DPN        |
           +-----------|-------------+        +----------|---------+
           |+----------v------------+|        |+---------v--------+|
           ||   [Data-plane model]  ||        ||[Data-plane model]||
           ||       FPC Agent       ||        ||    FPC Agent     ||
           |+-----------------------+|        |+------------------+|
           |+------------+----------+|        |                    |
           ||SB Protocol |FPC Client||        |  DPN Configuration |
           ||   Modules  |  Module  ||        +--------------------+
           |+------^-----+----^-----+|
           +-------|----------|------+
                   |          |
         Other     |          | FPC interface
        southbound |          | protocol
         protocols |          |
                   |          +-----------------+
                   |                            |
       DPN         |                 DPN        |
        +----------|---------+       +----------|---------+
        |+---------v--------+|       |+---------v--------+|
        ||  Configuration   ||       ||[Data-plane model]||
        || Protocol module  ||       ||     FPC Agent    ||
        |+------------------+|       |+------------------+|
        |                    |       |                    |
        | DPN Configuration  |       |  DPN Configuration |
        +--------------------+       +--------------------+
```

Figure 1: Reference Forwarding Policy Configuration (FPC) Architecture

The FPC architecture supports multi-tenancy; a FPC enabled data-plane supports tenants of multiple mobile operator networks and/or applications. It means that the FPC Client of each tenant connects to the FPC Agent and it MUST partition namespace and data for their data-planes. DPNs on the data-plane may fulfill multiple data-plane roles which are defined per session, domain and tenant.

Multi-tenancy permits the paritioning of data-plane entities as well as a common namespace requirement upon FPC Agents and Clients when they use the same Tenant for a common data-plane entity.

FPC information models often configuration to fit the specific needs for DPN management of a mobile node’s traffic. The FPC interfaces in Figure 1 are the only interfaces required to handle runtime data in a Mobility Context. The Topology and some Policy FPC models MAY be pre-configured; in that case real-time protocol exchanges are not required for them.

The information model provides an extensibility mechanism through Templates that permits specialization for the needs of a particular vendor’s equipment or future extension of the model presented in this specification.

4. FPC Mobility Information Model

The FPC information model includes the following components:

DPN Information Model,
Topology Information Model,
Policy Information Model,
Mobility-Context, and
Monitor, as illustrated in Figure 2.

```
                  :
                  |
                  +-[FPC Mobility Information Model]
                  |          |
                  |          +-[Topology Information Model]
                  |          |
                  |          +-[Policy Information Model]
                  |          |
                  |          +-[Mobility-Context]
                  |          |
                  |          +-[Monitor]
                  |
```

Figure 2: FPC Information Model structure

4.1. Model Notation and Conventions

The following conventions are used to describe the FPC information models.

Information model entities (e.g. DPNs, Rules, etc.) are defined in a hierarchical notation where all entities at the same hierarchical level are located on the same left-justified vertical position sequentially. When entities are composed of sub-entities, the sub-entities appear shifted to the right, as shown in Figure 3.

```
                           |
                           +-[entity2]
                           |         +-[entity2.1]
                           |         +-[entity2.2]
```

Figure 3: Model Notation - An Example

Some entities have one or more qualifiers placed on the right hand side of the element definition in angle-brackets. Common types include:

List: A collection of entities (some could be duplicated)

Set: A nonempty collection of entities without duplications

Name: A human-readable string

Key: A unique value. We distinguish 3 types of keys:

U-Key: A key unique across all Tenants. U-Key spaces typically involve the use of registries or language specific mechanisms that guarantee universal uniqueness of values.

G-Key: A key unique within a Tenant

L-Key: A key unique within a local namespace. For example, there may exist interfaces with the same name, e.g. "if0", in two different DPNs but there can only be one "if0" within each DPN (i.e. its local Interface-Key L-Key space).

Each entity or attribute may be optional (O) or mandatory (M). Entities that are not marked as optional are mandatory.

```
     The following example shows 3 entities:
         -- Entity1 is a globally unique key, and optionally can have
                    an associated Name
         -- Entity2 is a list
         -- Entity3 is a set and is optional
                    +
                    |
                    +-[entity1] <G-Key> (M), <Name> (O)
                    +-[entity2] <List>
                    +-[entity3] <Set> (O)
                    |
                    +
```

Figure 4

When expanding entity1 into a modeling language such as YANG it would result in two values: entity1-Key and entity1-Name.

To encourage re-use, FPC defines indexed sets of various entity Templates. Other model elements that need access to an indexed model entity contain an attribute which is always denoted as "entity-Key". When a Key attribute is encountered, the referencing model element may supply attribute values for use when the referenced entity model is instantiated. For example: Figure 5 shows 2 entities:

EntityA definition references an entityB model element.

EntityB model elements are indexed by entityB-Key.

Each EntityB model element has an entityB-Key which allows it to be uniquely identified, and a list of Attributes (or, alternatively, a Type) which specifies its form. This allows a referencing entity to create an instance by supplying entityB-Values to be inserted, in a Settings container.

```
                              .
                              .
                              |
                              +-[entityA]
                              |      +-[entityB-Key]
                              |      +-[entityB-Values]
                              .
                              .
                              |
                              +-[entityB] <L-Key> (M) <Set>
                              |      +-[entityB-Type]
                              .
                              .
```

Figure 5: Indexed sets of entities

Indexed sets are specified for each of the following kinds of entities:

```
Domain (See Section 4.9.3)
DPN (See Section 4.9.4)
Policy (See Section 4.9.5)
Rule (See Section 4.9.5)
Descriptor (See Figure 12)
Action (See Figure 12)
Service-Group (See Section 4.9.2, and
Mobility-Context (See Section 4.9.6)
```

As an example, for a Domain entity, there is a corresponding attribute denoted as "Domain-Key" whose value can be used to determine a reference to the Domain.

4.2. Templates and Attributes

In order to simplify development and maintenance of the needed policies and other objects used by FPC, the Information Models which are presented often have attributes that are not initialized with their final values. When an FPC entity is instantiated according to a template definition, specific values need to be configured for each such attribute. For instance, suppose an entity Template has an Attribute named "IPv4-Address", and also suppose that a FPC Client instantiates the entity and requests that it be installed on a DPN. An IPv4 address will be needed for the value of that Attribute before the entity can be used.

```
                        +-[Template] <U-Key, Name> (M) <Set>
                        |      +-[Attributes] <Set> (M)
                        |      +-[Extensible ~ FALSE]
                        |      +-[Entity-State ~ Initial]
                        |      +-[Version]
```

Figure 6: Template entities

Attributes: A set of Attribute names MAY be included when defining a Template for instantiating FPC entities.

Extensible: Determines whether or not entities instantiated from the Template can be extended with new non-mandatory Attributes not originally defined for the Template. Default value is FALSE. If a Template does not explicitly specify this attribute, the default value is considered to be in effect.

Entity-State: Either Initial, PartiallyConfigured, Configured, or Active. Default value is Initial. See Section 4.6 for more information about how the Entity-Status changes during the configuration steps of the Entity.

Version: Provides a version tag for the Template.

The Attributes in an Entity Template may be either mandatory or non-mandatory. Attribute values may also be associated with the attributes in the Entity Template. If supplied, the value may be either assigned with a default value that can be reconfigured later, or the value can be assigned with a static value that cannot be reconfigured later (see Section 4.3).

It is possible for a Template to provide values for all of its Attributes, so that no additional values are needed before the entity can made Active. Any instantiation from a Template MUST have at least one Attribute in order to be a useful entity unless the Template has none.

4.3. Attribute-Expressions

The syntax of the Attribute definition is formatted to make it clear. For every Attribute in the Entity Template, six possibilities are specified as follows:

'[Att-Name:]' Mandatory Attribute is defined, but template does not provide any configured value.

'[Att-Name: Att-Value]' Mandatory Attribute is defined, and has a statically configured value.

'[Att-Name: ˜ Att-Value]' Mandatory Attribute is defined, and has a default value.

'[Att-Name]' Non-mandatory Attribute may be included but template does not provide any configured value.

'[Att-Name = Att-Value]' Non-mandatory Attribute may be included and has a statically configured value.

'[Att-Name ˜ Att-Value]' Non-mandatory Attribute may be included and has a default value.

So, for example, a default value for a non-mandatory IPv4-Address attribute would be denoted by [IPv4-Address ˜ 127.0.0.1].

After a FPC Client identifies which additional Attributes have been configured to be included in an instantiated entity, those configured Attributes MUST NOT be deleted by the FPC Agent. Similarly, any statically configured value for an entity Attribute MUST NOT be changed by the FPC Agent.

Whenever there is danger of confusion, the fully qualified Attribute name MUST be used when supplying needed Attribute Values for a structured Attribute.

4.4. Attribute Value Types

For situations in which the type of an attribute value is required, the following syntax is recommended. To declare than an attribute has data type "foo", typecast the attribute name by using the parenthesized data type (foo). So, for instance, [(float) Max-Latency-in-ms:] would indicate that the mandatory Attribute "Max-Latency-in-ms" requires to be configured with a floating point value before the instantiated entity could be used. Similarly, [(float) Max-Latency-in-ms: 9.5] would statically configure a floating point value of 9.5 to the mandatory Attribute "Max-Latency-in-ms".

4.5. Namespace and Format

The identifiers and names in FPC models which reside in the same Tenant must be unique. That uniqueness must be maintained by all Clients, Agents and DPNs that support the Tenant. The Tenant namespace uniqueness MUST be applied to all elements of the tenant model, i.e. Topology, Policy and Mobility models.

When a Policy needs to be applied to Mobility-Contexts in all Tenants on an Agent, the Agent SHOULD define that policy to be visible by all Tenants. In this case, the Agent assigns a unique identifier in the

Agent namespace and copies the values to each Tenant. This effectively creates a U-Key although only a G-Key is required within the Tenant.

The notation for identifiers can utilize any format with agreement between data-plane agent and client operators. The formats include but are not limited to Globally Unique IDentifiers (GUIDs), Universally Unique IDentifiers (UUIDs), Fully Qualified Domain Names (FQDNs), Fully Qualified Path Names (FQPNs) and Uniform Resource Identifiers (URIs). The FPC model does not limit the format, which could dictate the choice of FPC protocol. Nevertheless, the identifiers which are used in a Mobility model should be considered to efficiently handle runtime parameters.

There are identifiers reserved for Protocol Operation. See Section 5.1.1.5 for details.

4.6. Configuring Attribute Values

Attributes of Information Model components such as policy templates are configured with values as part of FPC configuration operations. There may be several such configuration operations before the template instantiation is fully configured.

Entity-Status indicates when an Entity is usable within a DPN. This permits DPN design tradeoffs amongst local storage (or other resources), over the wire request size and the speed of request processing. For example, DPN designers with constrained systems MAY only house entities whose status is Active which may result in sending over all policy information with a Mobility-Context request. Storing information elements with an entity status of "PartiallyConfigured" on the DPN requires more resources but can result in smaller over the wire FPC communication and request processing efficiency.

When the FPC Client instantiates a Policy from a Template, the Policy-Status is "Initial". When the FPC Client sends the policy to a FPC Agent for installation on a DPN, the Client often will configure appropriate attribute values for the installation, and accordingly changes the Policy-Status to "PartiallyConfigured" or "Configured". The FPC Agent will also configure Domain-specific policies and DPN-specific policies on the DPN. When configured to provide particular services for mobile nodes, the FPC Agent will apply whatever service-specific policies are needed on the DPN. When a mobile node attaches to the network data-plane within the topology under the jurisdiction of a FPC Agent, the Agent may apply policies and settings as appropriate for that mobile node. Finally, when the mobile node launches new flows, or quenches existing flows, the FPC

Agent, on behalf of the FPC Client, applies or deactivates whatever policies and attribute values are appropriate for managing the flows of the mobile node. When a "Configured" policy is de-activated, Policy-Status is changed to be "Active". When an "Active" policy is activated, Policy-Status is changed to be "Configured".

Attribute values in DPN resident Policies may be configured by the FPC Agent as follows:

Domain-Policy-Configuration: Values for Policy attributes that are required for every DPN in the domain.

DPN-Policy-Configuration: Values for Policy attributes that are required for every policy configured on this DPN.

Service-Group-Policy-Configuration: Values for Policy attributes that are required to carry out the intended Service of the Service Group.

MN-Policy-Configuration: Values for Policy attributes that are required for all traffic to/from a particular mobile node.

Service-Data-Flow-Policy-Configuration: Values for Policy attributes that are required for traffic belonging to a particular set of flows on the mobile node.

Any configuration changes MAY also supply updated values for existing default attribute values that may have been previously configured on the DPN resident policy.

Entity blocks describe the format of the policy configurations.

4.7. Entity Configuration Blocks

As described in Section 4.6, a Policy Template may be configured in several stages by configuring default or missing values for Attributes that do not already have statically configured values. A Policy-Configuration is the combination of a Policy-Key (to identify the Policy Template defining the Attributes) and the currently configured Attribute Values to be applied to the Policy Template. Policy-Configurations MAY add attributes to a Template if Extensible is True. They MAY also refine existing attributes by:

assign new values if the Attribute is not static

make attributes static if they were not

make an attribute mandatory

A Policy-Configuration MUST NOT define or refine an attribute twice. More generally, an Entity-Configuration can be defined for any configurable Indexed Set to be the combination of the Entity-Key along with a set of Attribute-Expressions that supply configuration information for the entity's Attributes. Figure 7 shows a schematic representation for such Entity Configuration Blocks.

```
[Entity Configuration Block]
|       +-[Entity-Key] (M)
|       +-[Attribute-Expression] <Set> (M)
```

Figure 7: Entity Configuration Block

This document makes use of the following kinds of Entity Configuration Blocks:

Descriptor-Configuration

Action-Configuration

Rule-Configuration

Interface-Configuration

Service-Group-Configuration

Domain-Policy-Configuration

DPN-Policy-Configuration

Policy-Configuration

MN-Policy-Configuration

Service-Data-Flow-Policy-Configuration

4.8. Information Model Checkpoint

The Information Model Checkpoint permits Clients and Tenants with common scopes, referred to in this specification as Checkpoint BaseNames, to track the state of provisioned information on an Agent. The Agent records the Checkpoint BaseName and Checkpoint value set by a Client. When a Client attaches to the Agent it can query to determine the amount of work that must be executed to configure the Agent to a specific BaseName / checkpoint revision.

Checkpoints are defined for the following information model components:

Service-Group

DPN Information Model

Domain Information Model

Policy Information Model

4.9. Information Model Components

4.9.1. Topology Information Model

The Topology structure specifies DPNs and the communication paths between them. A network management system can use the Topology to select the most appropriate DPN resources for handling specific session flows.

The Topology structure is illustrated in Figure 8 (for definitions see Section 2):

```
                         |
                         +-[Topology Information Model]
                         |          +-[Extensible: FALSE]
                         |          +-[Service-Group]
                         |          +-[DPN] <Set>
                         |          +-[Domain] <Set>
```

Figure 8: Topology Structure

4.9.2. Service-Group

Service-Group-Set is collection of DPN interfaces serving some data-plane purpose including but not limited to DPN Interface selection to fulfill a Mobility-Context. Each Group contains a list of DPNs (referenced by DPN-Key) and selected interfaces (referenced by Interface-Key). The Interfaces are listed explicitly (rather than referred implicitly by its specific DPN) so that every Interface of a DPN is not required to be part of a Group. The information provided is sufficient to ensure that the Protocol, Settings (stored in the Service-Group-Configuration) and Features relevant to successful interface selection is present in the model.

```
   |
   +-[Service-Group] <G-Key>, <Name> (O) <Set>
   |            +-[Extensible: FALSE]
   |            +-[Role] <U-Key>
   |            +-[Protocol] <Set>
   |            +-[Feature] <Set> (O)
   |            +-[Service-Group-Configuration] <Set> (O)
   |            +-[DPN-Key] <Set>
   |            |            +-[Referenced-Interface] <Set>
   |            |            |        +-[Interface-Key] <L-Key>
   |            |            |        +-[Peer-Service-Group-Key] <Set> (O)
```

Figure 9: Service Group

Each Service-Group element contains the following information:

Service-Group-Key: A unique ID of the Service-Group.

Service-Group-Name: A human-readable display string.

Role: The role (MAG, LMA, etc.) of the device hosting the interfaces of the DPN Group.

Protocol-Set: The set of protocols supported by this interface (e.g., PMIP, S5-GTP, S5-PMIP etc.). The protocol MAY be only its name, e.g. ’gtp’, but many protocols implement specific message sets, e.g. s5-pmip, s8-pmip. When the Service-Group supports specific protocol message sub-subsets the Protocol value MUST include this information.

Feature-Set: An optional set of static features which further determine the suitability of the interface to the desired operation.

Service-Group-Configuration-Set: An optional set of configurations that further determine the suitability of an interface for the specific request. For example: SequenceNumber=ON/OFF.

DPN-Key-Set: A key used to identify the DPN.

Referenced-Interface-Set: The DPN Interfaces and peer Service-Groups associated with them. Each entry contains

Interface-Key: A key that is used together with the DPN-Key, to create a key that is refers to a specific DPN interface definition.

Peer-Service-Group-Key: Enables location of the peer Service-Group for this Interface.

4.9.3. Domain Information Model

A Domain-Set represents a group of heterogeneous Topology resources typically sharing a common administrative authority. Other models, outside of the scope of this specification, provide the details for the Domain.

```
             |
             +-[Domain] <G-Key>, <Name> (O) <Set>
             |       +-[Domain-Policy-Configuration] (O) <Set>
             |
```

Figure 10: Domain Information Model

Each Domain entry contains the following information:

Domain-Key: Identifies and enables reference to the Domain.

Domain-Name: A human-readable display string naming the Domain.

4.9.4. DPN Information Model

A DPN-Set contains some or all of the DPNs in the Tenant's network. Some of the DPNs in the Set may be identical in functionality and only differ by their Key.

```
               |
               +-[DPN] <G-Key>, <Name> (O) <Set>
               |     +-[Extensible: FALSE]
               |     +-[Interface] <L-Key> <Set>
               |     |      +-[Role] <U-Key>
               |     |      +-[Protocol] <Set>
               |     |      +-[Interface-Configuration] <Set> (O)
               |     +-[Domain-Key]
               |     +-[Service-Group-Key] <Set> (O)
               |     +-[DPN-Policy-Configuration] <List> (M)
               |     +-[DPN-Resource-Mapping-Reference] (O)
```

Figure 11: DPN Information Model

Each DPN entry contains the following information:

DPN-Key: A unique Identifier of the DPN.

DPN-Name: A human-readable display string.

Domain-Key: A Key providing access to the Domain information about the Domain in which the DPN resides.

Interface-Set: The Interface-Set references all interfaces (through which data packets are received and transmitted) available on the DPN. Each Interface makes use of attribute values that are specific to that interface, for example, the MTU size. These do not affect the DPN selection of active or enabled interfaces. Interfaces contain the following information:

Role: The role (MAG, LMA, PGW, AMF, etc.) of the DPN.

Protocol (Set): The set of protocols supported by this interface (e.g., PMIP, S5-GTP, S5-PMIP etc.). The protocol MAY implement specific message sets, e.g. s5-pmip, s8-pmip. When a protocol implements such message sub-subsets the Protocol value MUST include this information.

Interface-Configuration-Set: Configurable settings that further determine the suitability of an interface for the specific request. For example: SequenceNumber=ON/OFF.

Service-Group-Set: The Service-Group-Set references all of the Service-Groups which have been configured using Interfaces hosted on this DPN. The purpose of a Service-Group is not to describe each interface of each DPN, but rather to indicate interface types for use during the DPN selection process, when a DPN with specific interface capabilities is required.

DPN-Policy-Configuration: A list of Policies that have been configured on this DPN. Some may have values for all attributes, and some may require further configuration. Each Policy-Configuration has a key to enable reference to its Policy-Template. Each Policy-Configuration also has been configured to supply missing and non-default values to the desired Attributes defined within the Policy-Template.

DPN-Resource-Mapping-Reference (O): A reference to the underlying implementation, e.g. physical node, software module, etc. that supports this DPN. Further specification of this attribute is out of scope for this document.

4.9.5. Policy Information Model

The Policy Information Model defines and identifies Rules for enforcement at DPNs. A Policy is basically a set of Rules that are to be applied to each incoming or outgoing packet at a DPN interface. Rules comprise Descriptors and a set of Actions. The Descriptors, when evaluated, determine whether or not a set of Actions will be performed on the packet. The Policy structure is independent of a policy context.

In addition to the Policy structure, the Information Model (per Section 4.9.6) defines Mobility-Context. Each Mobility-Context may be configured with appropriate Attribute values, for example depending on the identity of a mobile node.

Traffic descriptions are defined in Descriptors, and treatments are defined separately in Actions. A Rule-Set binds Descriptors and associated Actions by reference, using Descriptor-Key and Action-Key. A Rule-Set is bound to a policy in the Policy-Set (using Policy-Key), and the Policy references the Rule definitions (using Rule-Key).

```
      |
      +-[Policy Information Model]
      |      +-[Extensible:]
      |      +-[Policy-Template] <G-Key> (M) <Set>
      |      |       +-[Policy-Configuration] <Set> (O)
      |      |       +-[Rule-Template-Key] <List> (M)
      |      |       |       +-[Precedence] (M)
      |      +-[Rule-Template] <L-Key> (M) <Set>
      |      |       +-[Descriptor-Match-Type] (M)
      |      |       +-[Descriptor-Configuration] <Set> (M)
      |      |       |       +-[Direction] (O)
      |      |       +-[Action-Configuration] <Set> (M)
      |      |       |       +-[Action-Order] (M)
      |      |       +-[Rule-Configuration] (O)
      |      +-[Descriptor-Template] <L-Key> (M) <Set>
      |      |       +-[Descriptor-Type] (O)
      |      |       +-[Attribute-Expression] <Set> (M)
      |      +-[Action-Template] <L-Key> (M) <Set>
      |              +-[Action-Type] (O)
      |      |       +-[Attribute-Expression] <Set> (M)
```

Figure 12: Policy Information Model

The Policy structure defines Policy-Set, Rule-Set, Descriptor-Set, and Action-Set, as follows:

Policy-Template: <Set> A set of Policy structures, indexed by Policy-Key, each of which is determined by a list of Rules referenced by their Rule-Key. Each Policy structure contains the following:

Policy-Key: Identifies and enables reference to this Policy definition.

Rule-Template-Key: Enables reference to a Rule template definition.

Rule-Precedence: For each Rule identified by a Rule-Template-Key in the Policy, specifies the order in which that Rule must be applied. The lower the numerical value of Precedence, the higher the rule precedence. Rules with equal precedence MAY be executed in parallel if supported by the DPN. If this value is absent, the rules SHOULD be applied in the order in which they appear in the Policy.

Rule-Template-Set: A set of Rule Template definitions indexed by Rule-Key. Each Rule is defined by a list of Descriptors (located by Descriptor-Key) and a list of Actions (located by Action-Key) as follows:

Rule-Template-Key: Identifies and enables reference to this Rule definition.

Descriptor-Match-Type Indicates whether the evaluation of the Rule proceeds by using conditional-AND, or conditional-OR, on the list of Descriptors.

Descriptor-Configuration: References a Descriptor template definition, along with an expression which names the Attributes for this instantiation from the Descriptor-Template and also specifies whether each Attribute of the Descriptor has a default value or a statically configured value, according to the syntax specified in Section 4.2.

Direction: Indicates if a rule applies to uplink traffic, to downlink traffic, or to both uplink and downlink traffic. Applying a rule to both uplink and downlink traffic, in case of symmetric rules, eliminates the requirement for a separate entry for each direction. When not present, the direction is implied by the Descriptor’s values.

Action-Configuration: References an Action Template definition, along with an expression which names the Attributes for this instantiation from the Action-Template and also specifies whether each Attribute of the Action has a default value or a statically configured value, according to the syntax specified in Section 4.2.

Action-Order: Defines the order in which actions are executed when the associated traffic descriptor selects the packet.

Descriptor-Template-Set: A set of traffic Descriptor Templates, each of which can be evaluated on the incoming or outgoing packet, returning a TRUE or FALSE value, defined as follows:

Descriptor-Template-Key: Identifies and enables reference to this descriptor template definition.

Attribute-Expression: An expression which defines an Attribute in the Descriptor-Template and also specifies whether the Template also defines a default value or a statically configured value for the Attribute of the Descriptor has, according to the syntax specified in Section 4.2.

Descriptor-Type: Identifies the type of descriptor, e.g. an IPv6 traffic selector per [RFC6088].

Action-Template-Set: A set of Action Templates defined as follows:

Action-Template-Key: Identifies and enables reference to this action template definition.

Attribute-Expression: An expression which defines an Attribute in the Action-Template and also specifies whether the Template also defines a default value or a statically configured value for the Attribute of the Action has, according to the syntax specified in Section 4.2.

Action-Type: Identifies the type of an action for unambiguous interpretation of an Action-Value entry.

4.9.6. Mobility-Context Information Model

The Mobility-Context structure holds entries associated with a mobile node and its mobility sessions (flows). It is created on a DPN during the mobile node’s registration to manage the mobile node’s flows. Flow information is added or deleted from the Mobility-Context as needed to support new flows or to deallocate resources for flows that are deactivated. Descriptors are used to characterize the nature and resource requirement for each flow.

Termination of a Mobility-Context implies termination of all flows represented in the Mobility-Context, e.g. after deregistration of a mobile node. If any Child-Contexts are defined, they are also terminated.

```
        +-[Mobility-Context] <G-Key> <Set>
        |            +-[Extensible:˜ FALSE]
        |            +-[Delegating-IP-Prefix:] <Set> (O)
        |            +-[Parent-Context] (O)
        |            +-[Child-Context] <Set> (O)
        |            +-[Service-Group-Key] <Set> (O)
        |            +-[Mobile-Node]
        |            |       +-[IP-Address] <Set> (O))
        |            |       +-[MN-Policy-Configuration] <Set>
        |            +-[Domain-Key]
        |            |       +-[Domain-Policy-Configuration] <Set>
        |            +-[DPN-Key] <Set>
        |            |       +-[Role]
        |            |       +-[DPN-Policy-Configuration] <Set>
        |            |       +-[ServiceDataFlow] <L-Key> <Set> (O)
        |            |       |       +-[Service-Group-Key] (O)
        |            |       |       +-[Interface-Key] <Set>
        |            |       |       +-[ServiceDataFlow-Policy-
                                           Configuration] <Set> (O)
        |            |       |       |       +-[Direction]
```

Figure 13: Mobility-Context Information Model

The Mobility-Context Substructure holds the following entries:

Mobility-Context-Key: Identifies a Mobility-Context

Delegating-IP-Prefix-Set: Delegated IP Prefixes assigned to the Mobility-Context

Parent-Context: If present, a Mobility Context from which the Attributes and Attribute Values of this Mobility Context are inherited.

Child-Context-Set: A set of Mobility Contexts which inherit the Attributes and Attribute Values of this Mobility Context.

Service-Group-Key: Service-Group(s) used during DPN assignment and re-assignment.

Mobile-Node: Attributes specific to the Mobile Node. It contains the following

IP-Address-Set IP addresses assigned to the Mobile Node.

MN-Policy-Configuration-Set For each MN-Policy in the set, a key and relevant information for the Policy Attributes.

Domain-Key: Enables access to a Domain instance.

Domain-Policy-Configuration-Set: For each Domain-Policy in the set, a key and relevant information for the Policy Attributes.

DPN-Key-Set: Enables access to a DPN instance assigned to a specific role, i.e. this is a Set that uses DPN-Key and Role as a compound key to access specific set instances.

Role: Role this DPN fulfills in the Mobility-Context.

DPN-Policy-Configuration-Set: For each DPN-Policy in the set, a key and relevant information for the Policy Attributes.

ServiceDataFlow-Key-Set: Characterizes a traffic flow that has been configured (and provided resources) on the DPN to support data-plane traffic to and from the mobile device.

> Service-Group-Key: Enables access to a Service-Group instance.
>
> Interface-Key-Set: Assigns the selected interface of the DPN.
>
> ServiceDataFlow-Policy-Configuration-Set: For each Policy in the set, a key and relevant information for the Policy Attributes.
>
> > Direction: Indicates if the reference Policy applies to uplink or downlink traffic, or to both, uplink- and downlink traffic. Applying a rule to both, uplink- and downlink traffic, in case of symmetric rules, allows omitting a separate entry for each direction. When not present the value is assumed to apply to both directions.

4.9.7. Monitor Information Model

Monitors provide a mechanism to produce reports when events occur. A Monitor will have a target that specifies what is to be watched.

The attribute/entity to be monitored places certain constraints on the configuration that can be specified. For example, a Monitor using a Threshold configuration cannot be applied to a Mobility-Context, because it does not have a threshold. Such a monitor configuration could be applied to a numeric threshold property of a Context.

```
                           |
                           +-[Monitor] <G-Key> <List>
                           |         +-[Extensible:]
                           |         +-[Target:]
                           |         +-[Deferrable]
                           |         +-[Configuration]
```

Figure 14: Monitor Substructure

Monitor-Key: Identifies the Monitor.

Target: Description of what is to be monitored. This can be a Service Data Flow, a Policy installed upon a DPN, values of a Mobility-Context, etc. The target name is the absolute information model path (separated by '/') to the attribute / entity to be monitored.

Deferrable: Indicates that a monitoring report can be delayed up to a defined maximum delay, set in the Agent, for possible bundling with other reports.

Configuration: Determined by the Monitor subtype. The monitor report is specified by the Configuration. Four report types are defined:

* "Periodic" reporting specifies an interval by which a notification is sent.

* "Event-List" reporting specifies a list of event types that, if they occur and are related to the monitored attribute, will result in sending a notification.

* "Scheduled" reporting specifies the time (in seconds since Jan 1, 1970) when a notification for the monitor should be sent. Once this Monitor's notification is completed the Monitor is automatically de-registered.

* "Threshold" reporting specifies one or both of a low and high threshold. When these values are crossed a corresponding notification is sent.

5. Protocol

5.1. Protocol Messages and Semantics

Four Client to Agent messages are supported.

Message	Description
Configure	A Configure message includes multiple edits to one or more information model entities. Edits are executed according to their Edit-Id in ascending order. The global status of the operation and the status of individual edits are returned. Partial failures, i.e. individual edit failures, are allowed.
Register-Monitors	Register monitors at an Agent. The message includes the Monitor information as specified in Section 4.9.7.
Deregister-Monitors	Deregister monitors from an Agent. An optional boolean, Send-Data, indicates if a successful deregistration triggers a Notify with final data from the Agent for the corresponding Monitor.
Probe	Probe the status of registered monitors. This triggers a Notify with current data from the Agent for the corresponding Monitors.

Table 1: Client to Agent Messages

Each message contains a header with the following information:

Client Identifier: An Identifier used by the Agent to associate specific configuration characteristics, e.g. options used by the Client when communicating with the Agent, the association of the Client and tenant in the information model as well as tracking operations and notifications.

Delay: An optional time (in ms) to delay the execution of the operation on the DPN once it is received by the Agent.

Operation Identifier: A unique identifier created by the Client to correlate responses and notifications

An Agent will respond with an ERROR, indicating one or more Errors have occurred, or an OK.

For Configure messages, an OK status for an edit MAY include subsequent edits in the response that were required to properly execute the edit. It MAY also indicate that the final status and any final edits required to fulfill the request will be sent via a

Configure Result Notification from the Agent to the Client, see Section 5.1.1.4.2.

If errors occur, they MUST be returned as a list in responses and each Error contains the following information:

Error-type: The specific error type. Values are TRANSPORT (0), RPC (1), PROTOCOL(2) or APPLICATION (3).

Error-Tag: An error tag.

Error-App-Tag: Application specific error tag.

Error-Message: A message describing the error.

Error-Info: Any data required for the response.

```
|
+-[Errors] <List>
|         +-[(Enumeration) Error-Type ]
|         +-[(String) Error-Tag ]
|         +-[(String) Error-App-Tag ] (O)
|         +-[(String) Error-Message ] (O)
|         +-[Error-Info] (O)
```

Figure 15: Error Information Model

Two Agent to Client notifications are supported.

Message	Description
Configure-Result-Notification	An asynchronous notification from Agent to Client based upon a previous Configure request.
Notify	An asynchronous notification from Agent to Client based upon a registered Monitor’s configuration, a Monitor deregistration or Probe.

Table 2: Agent to Client Messages (notifications)

5.1.1. Configure Message

The Configure message follows edit formats proposed by [RFC8072] with more fields in each edit, an extra operation (clone) and a different response format.

5.1.1.1. Edit Operation Types

Operation	Description
create	Creates a new data resource or Entity. If the resource exists an error is returned.
delete	Deletes a resource. If it does not exist an error is returned.
insert	Inserts data in a list or user ordered list.
merge	Merges the edit value with the target data resource; the resource is created if it does not exist.
move	Moves the target data resource.
replace	Replace the target data resource with the edit value.
remove	Removes a data resource if it already exists.
clone	Clones a data resource and places the copy at the new location. If the resource does not exist an error is returned.

Table 3: Configure Edit Operations

5.1.1.2. Edit Operation

Each Configure includes one or more edits. These edits include the following information:

Edit-Id: Uniquely specifies the identifier of the edit within the operation.

Edit-Type: Specifies the type of operation (see Section 5.1.1.1).

Command-Set: The Command-Set is a technology-specific bitset that allows for a single entity to be sent in an edit with multiple requested, technology specific sub-transactions to be completed. It can also provide clarity for a request. For example, a Mobility-Context could have the Home Network Prefix absent but it is unclear if the Client would like the address to be assigned by the Agent or if this is an error. Rather than creating a specific command for assigning the IP, a bit position in a Command-Set can be used to indicate Agent based IP assignment requests.

Reference-Scope: If supported, specifies the Reference Scope (see Section 5.1.1.3)

Target: Specifies the Target node (Data node path or FPC Identity) for the edit operation. This MAY be a resource, e.g. Mobility-Context, Descriptor-Template, etc., or a data node within a resource as specified by its path.

Point: The absolute URL path for the data node that is being used as the insertion point, clone point or move point for the target of this ’edit’ entry.

Where: Identifies where a data resource will be inserted, cloned to or moved. Only allowed these for lists and lists of data nodes that are ’ordered-by user’. The values are ’before’, ’after’, ’first’, ’last’ (default value).

Value The value used for this edit operation. In this message it MUST NOT be a MONITOR entity.

```
          |
          +-[Configure]
          |         +-[Client-Id:]
          |         +-[(Unsigned 32) Execution-Delay]
          |         +-[Operation-Id:]
          |         +-[Edit:] <List>
          |         |    +-[Edit-Id:] <L-Key>
          |         |    +-[(Enumeration) Edit-Type:]
          |         |    +-[(BitSet) Command-Set]
          |         |    +-[(Enumeration) Reference-Scope]
          |         |    +-[Target:]
          |         |    +-[Point]
          |         |    +-[(Enumeration) Where]
          |         |    +-[Value]
```

Figure 16: Configure Request

Edits sent to the Agent provided in an operation SHOULD be sent in the following order to avoid errors:

1. Action Templates

2. Descriptor Templates

3. Rule Templates

4. Policy Templates

5. DPN Templates

6. Mobility Contexts

5.1.1.3. Reference Scope

The Reference Scope is an optional feature that provides the scope of references used in a configuration command. These scopes are defined as:

- o none - All edits have no references to other entities or within edits.
- o edit - All references are contained within each edit body (intra-edit/intra-operation)
- o operation - All references exist in the operation (inter-edit/ intra-operation).
- o storage - One or more references exist outside of the operation. A lookup to cache / storage is required.
- o unknown - the location of the references are unknown. This is treated as a ’storage’ type.

An Agent that only accepts ’edit’ or ’operation’ reference scope messages is referred to as ’stateless’ as it has no direct memory of references outside messages themselves. This permits low memory footprint Agents/DPNs. Even when an Agent supports all message types an ’edit’ or ’operation’ scoped message can be processed quickly by the Agent/DPN as it does not require storage access.

Figure 17 shows an example containment hierarchy provided for all caches.

```
                         +---------------+
                         | Global Cache  |
                         |  (storage)    |
                         +------+--------+
                                |
                                +----------------+
                                |                |
                +------+-----------+      +------+-----------+
                | Operation Cache  |      | Operation Cache  |
                |   (operation)    | .... |   (operation)    |
                +------+-----------+      +--------+---------+
                       |                           |
                   +---+-----------+               |
                   |               |               |
            +------+------+ +------+------+ +------+------+
            | Edit Cache  | | Edit Cache  | | Edit Cache  |
            |   (edit)    | |   (edit)    | |   (edit)    |
            +-------------+ +-------------+ +-------------+

                                (no cache)
```

Figure 17: Example Hierarchical Cache

5.1.1.4. Operation Response

5.1.1.4.1. Immediate Response

The Response MUST include the following:

Operation Identifier of the corresponding request.

Global Status for the operation (see Table 1).

A list of Edit results (described below).

An edit response, Edit-Status, is comprised of the following:

Edit-Id: Edit Identifier.

Edit-Status: OK.

When the Edit-Status is OK the following values MAY be present

Notify-Follows - A boolean indicator that the edit has been accepted by the Agent but further processing is required. A Configure-Result-Notification will be sent once the processing has succeeded or failed.

      Subsequent-Edits-List: This is a list of Edits that were
      required to fulfill the request. It follows the edit request
      semantics (see Section 5.1.1.2).

   Errors-List: When the Edit-Status is ERROR the following values
   are present. See Table 1 for details.

The response will minimally contain an Edit-Status implying ’OK’ or a list of errors.

```
             |
             +-[Operation-Id:]
             +-[Result-Status:]
             +-[Errors] <List>
             |         +-[(Enumeration) Error-Type:]
             |         +-[(String) Error-Tag:]
             |         +-[(String) Error-App-Tag]
             |         +-[(String) Error-Message]
             |         +-[Error-Info]
             +-[Edit-Status]
             |         +-[Edit-Id:]
             |         +-[Edit-Status: ~ OK]
             |         +-[Notify-Follows]
             |         +-[Subsequent-Edits] <List>
             |         |    +-[Edit-Id:] <L-Key>
             |         |    +-[(Enumeration) Edit-Type:]
             |         |    +-[Target:]
             |         |    +-[Point]
             |         |    +-[(Enumeration) Where]
             |         |    +-[Value]
             |         +-[Errors] <List>
             |         |         +-[(Enumeration) Error-Type:]
             |         |         +-[(String) Error-Tag:]
             |         |         +-[(String) Error-App-Tag]
             |         |         +-[(String) Error-Message]
             |         |         +-[Error-Info]
             |
```

Figure 18: Configure Operation Response

5.1.1.4.2. Asynchronous Notification

A Configure-Result-Notification occurs after the Agent has completed processing related to a Configure request. It is an asynchronous communication from the Agent to the Client.

It is identical to the immediate response with the exception that the Notify-Follows, if present, MUST be false. As this value is unnecessary it SHOULD be omitted.

5.1.1.5. Reserved Identities

Several identities are reserved in the Policy Information Model and Mobility-Context to facilitate specific uses cases.

Agents and tenants express their support for descriptors and actions using the following Key patterns

supported-<descriptor template name> indicates a support for the descriptor template as defined in its original specification. For example "supported-rfc5777classifier" is a Descriptor Template that conforms to the rfc5777-classifier (Figure 31) as defined in this document.

supported-<action template name> indicates a support for the action template as defined in its original specification.

"base-rule" is comprised of all base descriptors using an ’or’ Descriptor-Match-Type and all Actions in no specific order.

"base-template" is comprised of the base rule.

"base-template" can be used to determine supported Action and Descriptor Templates. It can also be used to support an open template where any specific Descriptors and Actions can be applied, however, depending upon the Order of Actions it is likely to produce undesirable results.

One use case is supported via reservation of specific DPN-Keys:

Requested policies are those that the Client would like to be assigned to a DPN within a Mobility-Context. The naming convention is similar to those used for DPN Assignment via an Agent.

"Requested" is a Key that represents requested policies which have not been assigned to a specific DPN. No Role is assigned to the DPN.

"Requested-<Role>" represents requested policies that have not been assigned to a DPN and can only be assigned to DPNs that fulfill the specified Role.

It is possible to have policies in the "Requested" DPN that do not appear in other entries which reflects the inability to successfully assign the policy.

5.1.2. Monitor Messages

An Agent may reject a registration if it or the DPN has insufficient resources.

An Agent or DPN MAY temporarily suspend monitoring if insufficient resources exist. In such a case the Agent MUST notify the Client.

When a monitor has a reporting configuration of SCHEDULED it is automatically de-registered after the last Notify occurs.

If a SCHEDULED or PERIODIC configuration is provided during registration with the time related value (time or period respectively) of 0 a Notify is sent and the monitor is immediately de-registered. This method should, when a Monitor has not been installed, result in an immediate Notify sufficient for the Client’s needs and lets the Agent realize the Client has no further need for the monitor to be registered.

Probe messages are used by a Client to retrieve information about a previously installed monitor. The Probe message SHOULD identify one or more monitors by means of including the associated monitor identifier. An Agent receiving a Probe message sends the requested information in a single or multiple Notify messages.

If the Monitor configuration associated with a Notify can be deferred, then the Notify MAY be bundled with other messages back to the Agent even if this results in a delay of the Notify.

The Monitor messages use the following data:

Monitor-Key: Monitor Key.

Monitor: A Monitor configuration (see Section 4.9.7).

Send-Data: An indicator that specifies that the final value MUST be sent as a notification from the Agent.

```
             |
             +-[Register-Monitor]
             |         +-[Client-Id:]
             |         +-[(Unsigned 32) Execution-Delay]
             |         +-[Operation-Id:]
             |         +-[Monitor:] <List>
             |         |         +-[Extensible:]
             |         |         +-[Monitor-Key:] <U-Key>
             |         |         +-[Target:]
             |         |         +-[Deferrable]
             |         |         +-[Configuration:]

             |
             +-[Deregister-Monitor]
             |         +-[Client-Id:]
             |         +-[(Unsigned 32) Execution-Delay]
             |         +-[Operation-Id:]
             |         +-[Monitor:] <List>
             |         |         +-[Monitor-Key:] <U-Key>
             |         |         +-[(Boolean) Send-Data ˜ False]

             |
             +-[Probe]
             |         +-[Client-Id:]
             |         +-[(Unsigned 32) Execution-Delay]
             |         +-[Operation-Id:]
             |         +-[Monitor-Key:] <List>
```

Figure 19: Monitor Messages

5.1.2.1. Asynchronous Notification

A Monitor Report can be sent as part of de-registration, a trigger based upon a Monitor Configuration or a Probe. A Report is comprised of the Monitor Key the report applies to, the Trigger for the report, a timestamp of when the report’s associated event occurs and data, Report-Value, that is specific to the monitored value’s type.

Triggers include but are not limited to

o Subscribed Event occurred

o Low Threshold Crossed

o High Threshold Crossed

o Periodic Report

o Scheduled Report

o Probe

o Deregistration Final Value

o Monitoring Suspended

o Monitoring Resumed

o DPN Available

o DPN Unavailable

Multiple Reports are sent in a Notify message. Each Notify is comprised of unique Notification Identifier from the Agent and timestamp indicating when the notification was created.

```
        |
        +-[ Notify ]
        |         +-[(Unsigned 32) Notification-Identifier:]
        |         +-[Timestamp:]
        |         +-[Report:] <List>
        |         |    +-[Trigger:]
        |         |    +-[Monitor-Key:]
        |         |    +-[Report-Value]
```

Figure 20: Monitor Messages

5.2. Protocol Operation

Please note that JSON is used to represent the information in Figures in this section but any over the wire representation that accurately reflects the information model MAY be used.

5.2.1. DPN Selection

In order to assign a DPN to a Mobility Context, the Client or Agent requires topology information. The Service-Group provides information, e.g. function, role, protocol, features and configuration, to determine suitable DPN interfaces.

Consider a Client attempting to select DPN interfaces that are served by a single Agent. In this example interfaces are present with different protocols, settings and features as shown in the following figure.

```
 "topology-information-model" : {
     "dpn" : [ {
      "dpn-key" : "dpn1",
      "interface" : [ {
        "interface-key" : "ifc1",
        "role" : "lma",
        "protocol" : [ "pmip" ],
        "interface-configuration" : [ {
            "index" : 0,
            "setting" : [ "optionA" : "OFF" ]
        } ]
      },{
        "interface-key" : "ifc2",
        "role" : "lma",
        "protocol" : [ "pmip" ],
        "interface-configuration" : [ {
            "index" : 0,
            "setting" : [ "optionC" : "OFF"  ]
         } ]
       },{
        "interface-key" : "ifc2-b",
        "role" : "mag",
        "protocol" : [ "pmip" ]
        } ] },
      {
      "dpn-key" : "dpn2",
      "interface" : [ {
        "interface-key" : "ifc1",
        "role" : "mag",
        "protocol" : [ "pmip" ],
        "interface-configuration" : [ {
          "index" : 0,
          "settings" : [ "optionA" : "OFF", "optionB" : "ON" ]
        } ]
      } ] }
     ],
     ...
     },
     "service-group" : [
     { "service-group-key" : "group1",
       "service-group-name" : "Anchors-OptionA-OFF",
       "role-key" : "lma",
       "protocol" : [ "pmip" ],
       "service-group-configuration" : [ {
            "index" : 0,
            "setting" : [ "optionA" : "OFF" ]
       } ],
       "dpn" : [
       { "dpn-key" : "dpn1",
         "referenced-interface" : [ { "interface-key" : "ifc1" } ] }
       ]
     },{ "service-group-key" : "group2",
       "service-group-name" : "Anchors",
       "role-role" : "lma",
       "protocol" : [ "pmip" ],
       "dpn" : [
       { "dpn-key" : "dpn1",
         "referenced-interface" : [ { "interface-key" : "ifc2" } ] }
       ]
      },{ "service-group-key" : "group3",
       "service-group-name" : "MAGs",
       "role-role" : "mag",
       "protocol" : [ "pmip" ],
       "dpn" : [
       { "dpn-key" : "dpn2",
         "referenced-interface" : [ { "interface-key" : "ifc1" } ] },
       { "dpn-key" : "dpn1",
         "referenced-interface" : [ { "interface-key" : "ifc2-b" } ] }
       ]
      }
    ]
```

NOTE - A Setting is, in this example, a list of string attributes in a Configuration.

Figure 21: Monitor Messages

Two DPNs are present. The first, dpn1, has 3 interfaces. Two support the LMA role and both have settings. The third supports the MAG function. The second DPN, dpn2, provides a single interface with the MAG function.

Three ServiceGroups are presented. The first provides the PMIP protocol and LMA role. It also has a setting, OptionA, that is OFF and only contains ifc1 from dpn1.

The second group is comprised of interfaces that support the PMIP protocol and LMA function. It only contains ifc2 from dpn1. An interface that has setting(s) or feature(s) that must appear in a ServiceGroup SHOULD NOT appear in ServiceGroups that do not have those setting(s) or feature(s) present. Thus, ifc1 of dpn1 should not be present in this second Service-Group.

A third group is comprised of interfaces that support the MAG function of the LMA protocol. It contains the MAG interfaces form both dpn1 and dpn2.

Given the task to find a LMA that supports the PMIP protocol the Client can determine that dpn1 is its only option and, depending on its requirement of OptionA, can appropriately determine which interface to select.

5.2.2. Policy Creation and Installation

A Policy must be installed upon an Agent in order to install policies on the selected dpn(s). This requires construction of the Action(s), Descriptor(s) and Rule(s) used by the Policy.

The CONFIGURE message permits editing all information elements except monitors. The following figure shows use of a CONFIGURE message to install policy information on the Agent.

```
                                                 +-------Router--------+
                           +-----------+        |+-------+ +---------+|
    +------+ +------+     +-----+ FPC   |         | FPC   | |  Anchor |
    |MAG-C1| |MAG-C2|     |LMA-C| Client|         | Agent | |   DPN   |
    +------+ +------+     +-----+-------+         +-------+ +---------+
    |         |           |---(1)--Configure-------->|           |
    |         |          "configure" : {             |           |
    |         |           "client-id" : 0,           |           |
    |         |           "operation-id" : 0,        |           |
    |         |           "edit" : [ {               |           |
    |         |            "edit-id" : 0,            |           |
    |         |            "edit-type" : "create",   |           |
    |         |            "target" : "/policy-information-model/
                              /descriptor-template",              |
    |         |            "value" : {                           |
    |         |              "descriptor-template-key" : "desc1", |
    |         |              "descriptor-type" : "all-traffic" }  |
    |         |             }, {                                 |
    |         |            "edit-id" : 1,            |           |
    |         |            "edit-type" : "create",   |           |
    |         |            "target" : "/policy-information-model/
                              /action-template",                  |
    |         |            "value" : {                           |
    |         |              "action-template-key" : "action1",   |
    |         |              "action-type" : "drop" }             |
    |         |             }, {                                 |
    |         |            "edit-id" : 2,            |           |
    |         |            "edit-type" : "create",   |           |
    |         |            "target" : "/policy-information-model/
                              /rule-template",                    |
    |         |            "value" : {                           |
    |         |              "rule-template-key" : "deny-all",    |
    |         |              "descriptor-match-type" : "and",     |
    |          |              "descriptor-configuration" : [{       |
    |          |              "descriptor-template-key" : "all" }],|
    |          |              "action-configuration" : [{           |
    |          |                "action-template-key" : "deny",     |
    |          |                "action-order" : 0 }]               |
    |          |             }, {                                   |
    |          |            "edit-id" : 3,             |            |
    |          |            "edit-type" : "create",    |            |
    |          |            "target" : "/policy-information-model/
                              /policy-template",                   |
    |          |            "value" : {                             |
    |          |              "policy-template-key" : "policy1",    |
    |          |              "entity-state" : "configured",        |
    |          |              "rule-template" : [ {                 |
    |          |              "rule-template-key" : "deny-all",     |
    |          |              "precedence" : 0 } ]                  |
    |          |             } } ] }                                |
    |          |            |<---(2)- Response --------|            |
    |          |            | {                        |            |
    |          |            |  "agent-id" : "agent1"," |            |
    |          |            |  "operation-id" : 0,     |            |
    |          |            |  "result-status" : "ok"  |            |
    |          |            | }                        |            |
    |          |            |                          |            |
```

Figure 22: Example Policy Installation (focus on FPC reference point)

In this example a Descriptor "all-traffic" Template and an Action, "drop", Template are both empty Templates. The "deny-all" Rule Template is comprised of the action and descriptor. The Rule is included in "policy1". The policy’s status is "Configured" as it is a complete policy ready for immediate use. The policy could be set as "Active" if the Client intends to use it upon immediate installation in a DPN.

Installation of the policy on dpn1 is shown in the following Figure. The Policy-Status is set to "Active" to make it immediately usable. Leaving the status as Configured would permit its installation on the DPN without an ability to use it in a Mobility Context. Such a use case is often referred to as policy pre-configuration.

```
                                                 +-------Router--------+
                            +-----------+        |+-------+ +---------+|
    +------+ +------+     +-----+ FPC   |         | FPC   | |  Anchor |
    |MAG-C1| |MAG-C2|     |LMA-C| Client|         | Agent | |   DPN   |
    +------+ +------+     +-----+-------+         +-------+ +---------+
    |         |            |---(1)--Configure-------->|           |
    |         |          "configure" : {              |           |
    |         |           "client-id" : 0,            |           |
    |         |           "operation-id" : 1,         |           |
    |         |           "edit" : [ {                |           |
    |         |            "edit-id" : 0,             |           |
    |         |            "edit-type" : "create",    |           |
    |         |            "target" : "/topology-information-model/
                             /dpn/dpn1/dpn-policy-configuration", |
    |         |            "value" : {                            |
    |         |              "policy-template-key" : "policy1",   |
    |         |              "policy-status" : "active" } } ] }   |
    |         |            |<---(2)- Response --------|-policy--->|
    |         |            | {                        | install   |
    |         |            |  "agent-id" : "agent1"," |           |
    |         |            |  "operation-id" : 1,     |           |
    |         |            |  "result-status" : "ok"  |           |
    |         |            | }                        |           |
    |         |            |                          |           |
```

Figure 23: Example Policy Installation (focus on FPC reference point)

This message uses an edit type of "create" to add the policy template directly to the installed DPN policy set.

5.2.3. Simple RPC Operation

A Client and Agent MUST identify themselves using the Client Identifier and Agent Identifier respectively to ensure that, for all transactions, a recipient of a FPC message can unambiguously identify the sender of the FPC message.

A Client MAY direct the Agent to enforce a rule in a particular DPN by including a DPN Key value in a Mobility Context. Otherwise the Agent selects a suitable DPN to enforce one or more portions of a Mobility Context and notifies the Client about the selected DPN(s) using DPN Identifier(s).

All messages sent from a Client to an Agent MUST be acknowledged by the Agent. The response must include all edit status as well as subsequent edits, which indicates the result of processing the message, as part of the Configure response. In case the processing of the message results in a failure, the Agent sets the global status, Error-Type and Error-Tag accordingly and MAY clear the entity, e.g. Mobility-Context, which caused the failure, in the response.

If based upon Agent configuration or the processing of the request possibly taking a significant amount of time the Agent MAY respond with a Notify-Follows indication with optional Subsequent-Edit(s) containing the partially completed entity modifications. When a Notify-Follows indication is sent in a response, the Agent will, upon completion or failure of the operation, respond with an asynchronous Configuration-Result-Notification to the Client.

A Client MAY add a property to a Mobility-Context without providing all required details of the attribute’s value. In such case the Agent SHOULD determine the missing details and provide the completed property description, via Subsequent-Edit(s), back to the Client. If the processing will take too long or based upon Agent configuration, the Agent MAY respond with an OK for the Edit that indicates a Notify-Follows and also includes Subsequent-Edit(s) containing the partially completed entity edits.

In case the Agent cannot determine the missing value of an attribute’s value per the Client’s request, it leaves the attribute’s value cleared, sets the Edit Result to Error and provides an Error-Type and Error-Tag. As example, the Control-Plane needs to setup a tunnel configuration in the Data-Plane but has to rely on the Agent to determine the tunnel endpoint which is associated with the DPN that supports the Mobility-Context. The Client adds the tunnel property attribute to the FPC message and clears the value of the attribute (e.g. IP address of the local tunnel endpoint). The Agent determines the tunnel endpoint and includes the completed tunnel property in its response to the Client in a Subsequent-Edit entry.

Figure 24 illustrates an exemplary session life-cycle based on Proxy Mobile IPv6 registration via MAG Control-Plane function 1 (MAG-C1) and handover to MAG Control-Plane function 2 (MAG-C2). Edge DPN1 represents the Proxy CoA after attachment, whereas Edge DPN2 serves as Proxy CoA after handover. As exemplary architecture, the FPC Agent and the network control function are assumed to be co-located with the Anchor-DPN, e.g. a Router.

The Target of the second request uses the Mobility-Context by name. Alternatively, the Target could have included the DPN-Key and Policy-Key to further reduce the amount of information exchanged. Setting the Target’s value to the most specific node SHOULD be followed whenever practical.

```
                                          +-------Router--------+
                         +-----------+        |+-------+ +---------+|
   +------+ +------+     +-----+ FPC   |         | FPC    | |  Anchor |
   |MAG-C1| |MAG-C2|     |LMA-C| Client|         | Agent | |   DPN1  |
   +------+ +------+     +-----+-------+         +-------+ +---------+
   [MN attach]  |            |                          |           |
      |-------------PBU----->|                          |           |
      |         |            |---(1)--Configure-------->|           |
      |         |  "configure" : {                      |           |
      |         |    "client-id" : 0,                   |           |
      |         |    "operation-id" : 3,                |           |
      |         |    "edit" : [                         |           |
      |         |      "edit-id" : 0,                   |           |
      |         |      "edit-type" : "create",          |           |
      |         |      "target" : "/mobility-context",              |
      |         |      "value" : {                                  |
      |         |        "mobility-context-key" : "ctxt1",          |
      |         |        "delegating-ip-prefix" : [ <HNP> ],        |
      |         |        "dpn" : [ {                                |
      |         |          "dpn-key" : "DPN1",                      |
      |         |          "role" : "lma",                          |
      |         |          "service-data-flow" : [ {
      |         |            "identifier" : 0,
      |         |            "interface" : [ "interface-key" : "ifc1" ],
      |         |            "service-data-flow-policy-configuration" :[
      |         |              {"policy-template-key" :
      |         |                  "dl-tunnel-with-qos",
      |         |                "policy-status" : "active",
      |         |                "policy-configuration" : [
      |         |                  {"index" : 0,
      |         |                   "qos-template" : <QOS Settings...>},
      |         |                  {"index" : 1,
      |         |                   "tunnel" : <DL tunnel info...>},
      |         |               {"policy-template-key" : "ul-tunnel",
      |         |                   "policy-status" : "active",
      |         |                "policy-configuration" : [
      |         |                  {"index" : 1,
      |         |                   "tunnel" : <UL tunnel info...>}] }]
      |         |         } ] } ] } ] } ] }             |           |
      |         |          |                            |--tun1 up->|
      |         |          |                            |           |
      |         |          |                            |--tc qos-->|
      |         |          |                            |           |
      |         |          |<---(2)- Response --------|-route add>|
      |         |          | {                          |           |
      |         |          |  "agent-id" : "agent1","  |           |
      |         |          |  "operation-id" : 3,       |           |
      |         |          |  "result-status" : "ok",  |           |
      |         |          |  }                         |           |
      |         |            |                          |           |
      |<------------PBA------|                          |           |
      |         |            |                          |           |
      | +----+  |            |                          |           |
      | |Edge|  |            |                          |           |
      | |DPN1|  |            |                          |           |
      | +----+  |            |                          |           |
      |   |                                                         |
      |   |-=======================================================-|
      |                      |                          |           |
      |   [MN handover]      |                          |           |
      |         |---PBU ---->|                          |           |
      |         |            |--(3)- CONFIG(MODIFY)---->|           |
      |         |  "configure" : {                      |-tun1 mod->|
      |         |    "client-id" : 0,                   |           |
      |         |    "operation-id" : 4,                |           |
      |         |    "edit" : [                         |           |
      |         |      "edit-id" : 0,                   |           |
      |         |      "edit-type" : "merge",           |           |
      |         |      "target" : "/mobility-context/ctxt1",        |
      |         |      "value" : {                                  |
      |         |        "mobility-context-key" : "ctxt1",          |
      |         |        "dpn" : "[ {                               |
      |         |          "dpn-key" : "DPN1",                      |
      |         |          "service-data-flow" : [ {
      |         |            "identifier" : 0,
      |         |            "service-data-flow-policy-configuration":[
      |         |                {"policy-template-key" :
                                        "dl-tunnel-with-qos",
      |         |                "policy-configuration" : [
      |         |                  {"index" : 1,
      |         |                   "tunnel" : <NEW tunnel info...>}]}]
      |         |           } ] } ] } ] }               |           |
      |         |<--PBA------|                          |           |
      |         |            |                          |-tun1 mod->|
      |         |            |<---(4)- OK --------------|           |
      |         |            | {                        |           |
      |         |            |  "agent-id" : "agent1"," |           |
      |         |            |  "operation-id" : 4,     |           |
      |         |            |  "result-status" : "ok", |           |
      |         |            |  }                       |           |
      |         |  +----+    |                          |           |
      |         |  |Edge|    |                          |           |
      |         |  |DPN2|    |                          |           |
      |         |  +----+    |                          |           |
      |         |    |       |                          |           |
      |         |    |-============================================-|
      |         |            |                          |           |
```

Figure 24: Single Agent with Handover (focus on FPC reference point)

After reception of the Proxy Binding Update (PBU) at the LMA Control-Plane function (LMA-C), the LMA-C selects a suitable DPN, which serves as Data-Plane anchor to the mobile node’s (MN) traffic. The LMA-C adds a new logical Mobility-Context to the DPN to treat the MN’s traffic (1) and includes a Mobility-Context-Key (ctxt1) in the Configure command. The LMA-C identifies the selected Anchor DPN by including the associated DPN identifier.

The LMA-C adds policy template properties during the creation of the new Mobility-Context. One policy, "dl-tunnel-with-qos", is an example template that permits tunnel forwarding of traffic destined to the MN’s HNP, i.e. downlink traffic, with optional QoS parameters. Another policy, "ul-tunnel", provides a simple uplink anchor termination template where uplink tunnel information is provided.

The downlink tunnel information specifies the destination endpoint (Edge DPN1).

Upon reception of the Mobility-Context, the FPC Agent utilizes local configuration commands to create the tunnel (tun1) as well as the traffic control (tc) to enable QoS differentiation. After configuration has been completed, the Agent applies a new route to forward all traffic destined to the MN’s HNP specified as a property in the Mobility-Context and applied the configured tunnel interface (tun1).

During handover, the LMA-C receives an updating PBU from the handover target MAG-C2. The PBU refers to a new Data-Plane node (Edge DPN2) to represent the new tunnel endpoint in the downlink as required. The LMA-C sends a Configure message (3) to the Agent to modify the existing tunnel property of the existing Mobility-Context and to update the downlink tunnel endpoint from Edge DPN1 to Edge DPN2. Upon reception of the Configure message, the Agent applies updated tunnel property to the local configuration and responds to the Client (4).

```
                                               +-------Router--------+
                        +-----------+          |+-------+ +---------+|
+------+ +------+     +-----+ FPC   |           | FPC   | |  Anchor |
|MAG-C1| |MAG-C2|     |LMA-C| Client|           | Agent | |   DPN   |
+------+ +------+     +-----+-------+           +-------+ +---------+
[MN detach]  |            |                           |           |
   |-------------PBU----->|                           |           |
   |         |            |---(1)--Configure--------->|           |
   |         |          "configure" : {             |           |
   |         |           "client-identifier" : 0,    |           |
   |         |           "operation-id" : 5,          |           |
   |         |           "edits" : [                  |           |
   |         |            "edit-id" : 0,              |           |
   |         |            "edit-type" : "merge",      |           |
   |         |            "target" : "/mobility-context/ctxt1     |
   |         |                /dpn/DPN1/service-data-flow/0       |
   |         |                /service-data-flow-policy-
                               configuration/dl-tunnel-with-qos/1"
   |         |            "value" : {                 |           |
   |         |                  "tunnel" : null       |           |
   |         |             } ] }                      |           |
   |<------------PBA------|                           |--tun1   ->|
   |         |            |                           |    down   |
   |         |            |                           |           |
   |         |            |<---(2)- Response --------|           |
   |         |            | {                         |           |
   |         |            |  "agent-id" : "agent1","  |           |
   |         |            |  "operation-id" : 5,      |           |
   |         |            |  "result-status" : "ok",  |           |
   |         |            | }                         |           |
   |         |            |                           |           |
   |         |  [ MinDelayBeforeBCEDelete expires ]   |           |
   |         |            |                           |           |
   |         |            |---(3)--Configure-------->|-- tun1 -->|
   |         |          "configure" : {               |  delete   |
   |         |           "client-identifier" : 0,     |           |
   |         |           "operation-id" : 6,          |           |
   |         |           "edits" : [                  |           |
   |         |            "edit-id" : 0,              |           |
   |         |            "edit-type" : "delete",     |           |
   |         |            "target" : "/mobility-context/ctxt1"    |
   |         |            ] }                         |           |
   |         |            |                           |           |
   |         |            |<---(4)- Response --------|           |
   |         |            | {                         |           |
   |         |            |  "agent-id" : "agent1","  |           |
   |         |            |  "operation-id" : 6,      |           |
   |         |            |  "result-status" : "ok",  |           |
   |         |            | }                         |           |
   |         |            |                           |-- route ->|
   |         |            |                           |   remove  |
   |         |            |                           |           |
```

Figure 25: Single Agent with Deletion (focus on FPC reference point)

When a teardown of the session occurs, MAG-C1 will send a PBU with a lifetime value of zero. The LMA-C sends a Configure message (1) to the Agent to modify the existing tunnel property of the existing

Mobility-Context to delete the tunnel information. Upon reception of the Configure message, the Agent removes the tunnel configuration and responds to the Client (2). Per [RFC5213], the PBA is sent back immediately after the PBA is received.

If no valid PBA is received after the expiration of the MinDelayBeforeBCEDelete timer (see [RFC5213]), the LMA-C will send a Configure (3) message with a deletion request for the Context. Upon reception of the message, the Agent deletes the tunnel and route on the DPN and responds to the Client (4).

When a multi-DPN Agent is used the DPN list permits several DPNs to be provisioned in a single message for the single Mobility-Context.

```
                           +-----------+           +-------+ +---------+
   +------+ +------+     +-----+ FPC   |           | FPC   | |  Anchor |
   |MAG-C1| |MAG-C2|     |LMA-C| Client|           | Agent | |   DPN1  |
   +------+ +------+     +-----+-------+           +-------+ +---------+
   [MN attach]  |            |                          |           |
      |-------------PBU----->|                          |           |
      |         |            |---(1)--Configure-------->|           |
      |         |  "configure" : {                      |--tun1 up->|
      |         |    "client-id" : 0,                   |           |
      |         |    "operation-id" : 0,                |           |
      |         |    "edit" : [                         |--tc qos-->|
      |         |      "edit-id" : 0,                   |           |
      |         |      "edit-type" : "create",          |           |
      |         |      "target" : "/mobility-context",              |
      |         |      "value" : {                                  |
      |         |        "mobility-context-key" : "ctxt1",          |
      |         |        "delegating-ip-prefix" : [ <HNP> ],        |
      |         |        "dpn" : [ {                                |
      |         |          "role" : "lma",                          |
      |         |          "dpn-key" : "DPN1",                      |
      |         |          "service-data-flow" : [ {
      |         |            "identifier" : 0,
      |         |            "interface" : [ "interface-key" : "ifc1" ],
      |         |            "service-data-flow-policy-configuration" :[
      |         |              {"policy-template-key" :
                                        "dl-tunnel-with-qos",
      |         |                "policy-status" : "active",
      |         |                "policy-configuration" : [
      |         |                  {"index" : 0,
      |         |                   "qos-template" : <QOS Settings...>},
      |         |                  {"index" : 1,
      |         |                   "tunnel" : <DL tunnel info...>},
      |         |               {"policy-template-key" : "ul-tunnel",
      |         |                   "policy-status" : "active",
      |         |                "policy-configuration" : [
      |         |                  {"index" : 1,
      |         |                   "tunnel" : <UL tunnel info...>}] }]
      |         |              } ] } ] }, {
      |         |          "dpn-key" : "DPN2",                        |
      |         |          "role" : "mag",                            |
      |         |          "service-data-flow" : [ {
      |         |            "identifier" : 0,
      |         |            "interface" : [ "interface-key" : "ifc2" ],
      |         |            "service-data-flow-policy-configuration" :[
      |         |              {"policy-template-key" :
      "dl-tunnel-with-qos",
      |         |                "policy-status" : "active",
      |         |                "policy-configuration" : [
      |         |                  {"index" : 0,
      |         |                   "qos-template" : <QOS Settings...>},
      |         |                  {"index" : 1,
      |         |                   "tunnel" : <DL tunnel info...>},
      |         |               {"policy-template-key" : "ul-tunnel",
      |         |                   "policy-status" : "active",
      |         |                "policy-configuration" : [
      |         |                  {"index" : 1,
      |         |                   "tunnel" : <UL tunnel info...>}] }]
      |         |              } ] } ] }                    |           |
      |         |              ] } ] }                      |           |
      |         |            |                              |           |
      |         |            |<---(2)- Response --------|           |
      |         |            | {                            |-route add>|
      |         |            |  "agent-id" : "agent1","  |           |
      |         |            |  "operation-id" : 0,         |           |
      |         |            |  "result-status" : "ok",  |           |
      |         |            |  "notify-follows" : "true",           |
      |         |            | }                            |           |
      |         |            |                              |           |
      |<------------PBA------|                              |           |
      |         |            |                              |           |
      | +----+               |                              |           |
      | |Edge|               |                              |           |
      | |DPN2|               |                              |           |
      | +----+               |                              |           |
      |   |<--------------------- tun1 up -------------|           |
      |   |<--------------------- tc qos --------------|           |
      |   |<--------------------- route add -----------|           |
      |   |                  |                              |           |
      |         |            |<(3) Configure-Result-    |           |
      |         |            |       Notification           |           |
      |         |            | {                            |-route add>|
      |         |            |  "agent-id" : "agent1","  |           |
      |         |            |  "operation-id" : 3,     |            |
      |         |            |  "result-status" : "ok", |            |
      |         |            |  "notify-follows" : "true",            |
      |         |            |  "edit-status" : [       |            |
      |         |            |    "edit-id" : 0,        |            |
      |         |            |    "edit-status" : "ok"  |            |
      |         |            | ] }                      |            |
      |         |            |                          |            |
      |   |                  |                          |            |
```

Figure 26: Exemplary Message Sequence for Multi-DPN Agent

Figure 26 shows how the first 2 messages in Figure 24 are supported when a multi-DPN Agent communicates with both Anchor DPN1 and Edge DPN2. In such a case, the FPC Client sends the downlink and uplink for both DPNs in the DPN Reference List of the same Mobility-Context. Message 1 shows the DPN Set with all entries. Each entry identifies the DPN.

The Agent responds with an OK and Notify-Follows indication while it simultaneously provisions both DPNs. Upon successful completion, the Agent responds to the Client with a Configuration-Result-Notification indicating the operation status.

5.2.4. Policy and Mobility on the Agent

A Client may build Policy and Topology using Configure messages.

The Client may add, modify or delete many DPN Policies as DPN Policy Configurations and Mobility-Contexts in a single FPC message. This includes linking Mobility-Contexts to DPN Policies as well as creating the Policy, Rules Actions and Descriptors. As example, a Rule which performs re-writing of an arriving packet’s destination IP address from IP_A to IP_B matching an associated Descriptor, can be enforced in the Data-Plane via an Agent to implicitly consider matching arriving packet’s source IP address against IP_B and re-write the source IP address to IP_A.

Figure 27 illustrates the generic policy configuration model as used between a FPC Client and a FPC Agent.

```
      Descriptor_1 -+          +- Action_1
                   |          |
      Descriptor_2 -+--<Rule>--+- Action_2
                     +-----------+
                     /Precendent#/--------+
                     +----------+         |
                                          |
      Descriptor_3 -+          +- Action_3 +-<Policy>
                   |          |           |  ^
      Descriptor_4 -+--<Rule>--+- Action_4 |  |
                     +-----------+         |  |
                     /Precendent#/--------+  |
                     +----------+            |
                                           <DPN-Policy-Configuration>

      +---------------------+     +----------------------+
      | Bind 1..M traffic   |     |  Bind 1..N traffic   |
      |  Descriptors to     | --> |  treatment actions   |
      |  to a Policy        |     |   to a Policy        |
      +---------------------+     +----------------------+

     |                                                  |
     +-------------- Data-Plane Rule ------------------+
```

Figure 27: Structure of Configurable Policies

As depicted in Figure 27, the DPN Settings represents the anchor of Rules through the Policy / Rule hierarchy. A Client and Agent use the identifier of the associated Policy to directly access the Rule and perform modifications of traffic Descriptors or Action references. Arriving packets are matched against traffic according to Rule precedence and Descriptors. If a Rule is applicable the packet is treated according to the ordered Action values.

A Client associates a Precedence value for the Rule’s Descriptors, to allow unambiguous traffic matching on the Data-Plane.

Figure 28 illustrates the generic context configuration model as used between a Client and an Agent.

```
             <Policy 1>
                   ^
                   |
             <Service-Data-Flow 0> <--- <Mobility-Context-ID2>
                                                  ^
                                                  |
             <Policy 1>                           |
             ^                                    |
             |                                    |
             <Service-Data-Flow 0> <--- <Mobility-Context-ID1>

             +-------------------+     +---------------------+
             | Bind 1..M traffic |     |  Bind 1..N traffic  |
             |    selectors to   | --> |  treatment / qos    |
             |     a Context     |     |  actions to a       |
             |                   |     |       Context       |
             +-------------------+     +---------------------+

            |                                                  |
            +-------------- Data-Plane Rule ------------------+
```

Figure 28: Mobility Context Hierarchy

Figure 28 represents a mobility session hierarchy. A Client and Agent directly assigns values such as downlink traffic descriptors, QoS information, etc. A Client and Agent use the context identifiers to access the descriptors, qos information, etc. to perform modifications. From the viewpoint of packet processing, arriving packets are matched against traffic Descriptors and processed according to the qos or other mobility profile related Actions specified in the Mobility-Context’s and Service-Data-Flow’s’ properties. If present, a Policy could contain tunnel information to encapsulate and forward the packet.

A second Mobility-Context also references Mobility-Context-ID1 in the figure. Based upon the technology a property in a parent context (parent mobility-context-id reference) MAY be inherited by its descendants. This permits concise over the wire representation. When a Client deletes a parent Context all children are also deleted.

5.2.5. Monitor Example

The following example shows the installation of a DPN level monitor (1) to observe ifc1 status, a property that is either "up" or "down", and another monitor to watch for interface events. The interface experiences an outage which is reported to the Client via a Notify (3) message. At a later time a Probe (4) and corresponding Notify (5) is sent. Finally, the monitors are de-registered (6).

   Note, specific event identifiers and types are out of scope.

```
                                                +-------Router--------+
                          +-----------+        |+-------+ +---------+|
     +------+ +------+   +-----+ FPC   |         | FPC   | |  Anchor |
     |MAG-C1| |MAG-C2|   |LMA-C| Client|         | Agent | |   DPN   |
     +------+ +------+   +-----+-------+         +-------+ +---------+
     |         |          |---(1)--Configure-------->|          |
     |         |       "register-monitor" : {        |          |
     |         |         "client-id" : 0,            |          |
     |         |         "operation-id" : 0,         |          |
     |         |         "monitor" : [ {             |          |
     |         |          "monitor-key" : "ifc1-status",        |
     |         |          "target" : "/dpn/dpn1/interface/ifc1/status"
     |         |          "deferrable" : false                  |
     |         |          }, {                                  |
     |         |          "monitor-key" : "ifc1-events",        |
     |         |          "target" : "/dpn/dpn1/interface/ifc1" |
     |         |          "deferrable" : false,                 |
     |         |          "configuration" : {                   |
     |         |              "target-event-configuration" : [ 0,
     |         |                1, 3, .. ] } } ] }              |
     |         |          |<---(2)- Response --------|          |
     |         |          | {                        |          |
     |         |          |  "agent-id" : "agent1"," |          |
     |         |          |  "operation-id" : 0,     |          |
     |         |          |  "result-status" : "ok", |          |
     |         |          | }                        |          |
     |         |          |                          |          |
               [ ifc1 goes down which is reported as event type 3 ]
     |         |          |<---(3)-- NOTIFY ---------|          |
     |         |        "notify" : {                 |          |
     |         |         "notification-id" : 0,      |          |
     |         |         "timestamp" : ...,          |          |
     |         |         "report" : [ {              |          |
     |         |          "monitor-key" : "ifc1-events",        |
     |         |          "trigger" : "subscribed-event-occurred",
     |         |          "report-value" : { 3 } } ] }          |
     |         |          |                          |          |
     |         |          |---(4)--  Probe  -------->|          |
     |         |        "probe" : {      |           |
     |         |         "client-id" : 0,            |          |
     |         |         "operation-id" : 1,         |          |
     |         |         "monitor" : [               |          |
     |         |          "monitor-key" : "ifc1-status" ] }     |
     |         |          |<---(5)- Response --------|          |
     |         |          | {                        |          |
     |         |          |  "agent-id" : "agent1"," |          |
     |         |            |  "operation-id" : 1,     |           |
     |         |            |  "result-status" : "ok", |           |
     |         |            | }                        |           |
     |         |            |                          |           |
     |         |            |<---(6)-- NOTIFY ---------|           |
     |         |          "notify" : {                 |           |
     |         |           "notification-id" : 1,      |           |
     |         |           "timestamp" : ...,          |           |
     |         |           "report" : [ {              |           |
     |         |            "monitor-key" : "ifc1-status",         |
     |         |            "trigger" : "probe",
     |         |            "report-valuerporte" : { "up" } } ] }  |
     |         |            |                          |           |
     |         |            |---(7)- Deregister ------>|           |
     |         |          "deregister-monitor" : {     |           |
     |         |           "client-id" : 0,            |           |
     |         |           "operation-id" : 2,         |           |
     |         |           "monitor" : [               |           |
     |         |            { "monitor-key" : "ifc1-events" },     |
     |         |            { "monitor-key" : "ifc1-status",       |
     |         |              "send-data" : true } ] }             |
     |         |            |<---(8)- Response --------|           |
     |         |            | {                        |           |
     |         |            |  "agent-id" : "agent1"," |           |
     |         |            |  "operation-id" : 2,     |           |
     |         |            |  "result-status" : "ok", |           |
     |         |            | }                        |           |
     |         |            |                          |           |
     |         |            |<---(9)-- NOTIFY ---------|           |
     |         |          "notify" : {                 |           |
     |         |           "notification-id" : 2,      |           |
     |         |           "timestamp" : ...,          |           |
     |         |           "report" : [ {              |           |
     |         |            "monitor-key" : "ifc1-status",         |
     |         |            "trigger" : "deregistration-final-value",
     |         |            "report-value" : { "up" } } ] }        |
```

Figure 29: Monitor Example (focus on FPC reference point)

6. Templates and Command Sets

Configuration templates are shown below.

6.1. Monitor Configuration Templates

A periodic configuration specifies a time interval (ms) for reporting.

A scheduled configuration specifies a time for reporting.

A threshold configuration MUST have at least one hi or low threshold and MAY have both.

A Target-Events-Configuration is a list of Events that, when generated by the Target, results in a Monitor notification.

```
     |
     +-[Monitor] <List>
     ...
     |         +-[Configuration]
     |         |     +-[Periodic-Configuration]
     |         |     |       +-[(Unsigned32) Period:]
     ...
     |         +-[Configuration]
     |         |     +-[Schedule-Configuration]
     |         |     |       +-[(Unsigned32) Schedule:]
     ...
     |         +-[Configuration]
     |         |     +-[Threshold-Configuration]
     |         |     |       +-[(Unsigned32) Low]
     |         |     |       +-[(Unsigned32) Hi]
     ...
     |         +-[Configuration]
     |         |     +-[Target-Events-Configuration]
     |         |     |       +-[(Unsigned32) Event-Key:] <List>
```

Figure 30: Monitor Configuration Templates

6.2. Descriptor Templates

A IP-Prefix-Template MUST have at least the To or From IP Prefix / Length populated. The IP Prefix specifies and Address and Length.

The PMIP Traffic Selector template is mapped according to [RFC6088]

The RFC 5777 Classifier is a structured version of common filter rules and follows the format specified in [RFC5777]. The Flow-Label, Flow-Label range and ECN-IP-Codepoint specified in [RFC7660] are added to the Descriptor as well.

```
  |
  +-[ip-prefix-template]
  |       +-[(IP Prefix / Length) To-IP-Prefix]
  |       +-[(IP Prefix / Length) From-IP-Prefix]
  ...
  +-[pmip-traffic-selector]
```

```
     |       +-[(Enumerated - IPv4 or IPv6) ts-format]
     |       +-[ipsec-spi-range]
     |       |   +-[ (ipsec-spi) start-spi: ]
     |       |   +-[ (ipsec-spi) end-spi ]
     |       +-[source-port-range]
     |       |   +-[ (port-number) start-port: ]
     |       |   +-[ (port-number) end-port ]
     |       +-[destination-port-range]
     |       |   +-[ (port-number) start-port: ]
     |       |   +-[ (port-number) end-port ]
     |       +-[source-address-range-v4]
     |       |   +-[ (ipv4-address) start-address: ]
     |       |   +-[ (ipv4-address) end-address ]
     |       +-[destination-address-range-v4]
     |       |   +-[ (ipv4-address) start-address: ]
     |       |   +-[ (ipv4-address) end-address ]
     |       +-[ds-range]
     |       |   +-[ (dscp) start-ds: ]
     |       |   +-[ (dscp) end-ds ]
     |       +-[protocol-range]
     |       |   +-[ (uint8) start-protocol: ]
     |       |   +-[ (uint8) end-protocol ]
     |       +-[source-address-range-v6]
     |       |   +-[(ipv6-address) start-address: ]
     |       |   +-[(ipv6-address) end-address ]
     |       +-[destination-address-range-v6]
     |       |   +-[(ipv6-address) start-address: ]
     |       |   +-[(ipv6-address) end-address ]
     |       +-[flow-label-range]
     |       |   +-[(ipv6-flow-label) start-flow-label ]
     |       |   +-[(ipv6-flow-label) end-flow-label ]
     |       +-[traffic-class-range]
     |       |   +-[ (dscp) start-traffic-class ]
     |       |   +-[ (dscp) end-traffic-class ]
     |       +-[next-header-range]
     |       |   +-[ (uint8) start-next-header ]
     |       |   +-[ (uint8) end-next-header ]
     ...
     +-[rfc5777-classifier]
     |       +-[Extensible: True]
     |       +-[(uint8) protocol]
     |       +-[(Enumerated - In/Out/Both) Direction]
     |       +-[From-Spec] <List>
     |       |    +-[(ip-address) IP-Address] <List>
     |       |    +-[IP-Address-Range] <List>
     |       |    |    +-[(ip-address) IP-Address-Start]
     |       |    |    +-[(ip-address) IP-Address-End]
     |       |    +-[IP-Address-Mask] <List>
     |       |    |    +-[(ip-address) IP-Address:]
     |       |    |    +-[(Unsigned 32) IP-Bit-Mask-Width:]
     |       |    +-[(mac-address) MAC-Address] <List>
     |       |    +-[MAC-Address-Mask] <List>
     |       |    |    +-[(mac-address) MAC-Address:]
     |       |    |    +-[(mac-address) MAC-Address-Mask-Pattern:]
     |       |    +-[(eui64-address) EUI64-Address] <List>
     |       |    +-[EUI64-Address-Mask] <List>
     |       |    |    +-[(eui64-address) EUI64-Address:]
     |       |    |    +-[(eui64-address) EUI64-Address-Mask-Pattern:]
     |       |    +-[(Integer 32) Port] <List>
     |       |    +-[Port-Range] <List>
     |       |    |    +-[(Integer 32) Port-Start]
     |       |    |    +-[(Integer 32) Port-End]
     |       |    +-[(Boolean) Negated]
     |       |    +-[(Boolean) Use-Assigned-Address]
     |       +-[To-Spec] <List> (O)
     |       |    +-[(ip-address) IP-Address] <List>
     |       |    +-[IP-Address-Range] <List>
     |       |    |    +-[(ip-address) IP-Address-Start]
     |       |    |    +-[(ip-address) IP-Address-End]
     |       |    +-[IP-Address-Mask] <List>
     |       |    |    +-[(ip-address) IP-Address:]
     |       |    |    +-[(Unsigned 32) IP-Bit-Mask-Width:]
     |       |    +-[(mac-address) MAC-Address] <List>
     |       |    +-[MAC-Address-Mask] <List>
     |       |    |    +-[(mac-address) MAC-Address:]
     |       |    |    +-[(mac-address) MAC-Address-Mask-Pattern:]
     |       |    +-[(eui64-address) EUI64-Address] <List>
     |       |    +-[EUI64-Address-Mask] <List>
     |       |    |    +-[(eui64-address) EUI64-Address:]
     |       |    |    +-[(eui64-address) EUI64-Address-Mask-Pattern:]
     |       |    +-[(Integer 32) Port] <List>
     |       |    +-[Port-Range] <List>
     |       |    |    +-[(Integer 32) Port-Start]
     |       |    |    +-[(Integer 32) Port-End]
     |       |    +-[(Boolean) Negated]
     |       |    +-[(Boolean) Use-Assigned-Address]
     |       +-[(dscp) Diffserv-Code-Point] <List>
     |       +-[(Boolean) Fragmentation-Flag ~ False]
     |       +-[IP-Option] <List>
     |       +-[TCP-Option] <List>
     |       +-[TCP-Flags]
     |       +-[ICMP-Type] <List>
     |       +-[ETH-Option] <List>
     |       +-[ecn-ip-codepoint] <List>
     |       +-[(flowlabel) flow-label] <List>
     |       +-[flow-label-range] <List>
     |       |    +-[(flowlabel) flow-label-start]
     |       |    +-[(flowlabel) flow-label-end]
```

Figure 31: Descriptor Templates

6.3. Tunnel Templates

The Network Service Header is specified in [RFC8300].

The MPLS SR Stack is specified in [I-D.ietf-spring-segment-routing-mpls].

The IPv6 SR Stack is specified in [I-D.ietf-6man-segment-routing-header].

A tunnel MUST have the local-address or remote-address (or both) populated.

For GRE, the gre-key MUST be present.

For GTP (GPRS Tunneling Protocol), the following attributes MAY be present

local tunnel endpoint identifier (teid) - MUST be present if local-address is nonempty

remote tunnel endpoint identifier (teid) - MUST be present if remote-address is nonempty

sequence-numbers-on - Indicates that sequence numbers will be used

Tunnels can be used as Next Hop and Descriptor values.

```
     |
     +-[next-hop-template]
     |     +-[Extensible: True]
     |     +-[(ip-address) address]
     |     +-[(mac-address) mac-address]
     |     +-[(service-path-id) service-path]
     |     +-[(mpls-label) mpls-path]
     |     +-[(network service header) nsh]
     |     +-[(Unsigned Integer) interface]
     |     +-[(Unsigned 128) segment-identifier]
     |     +-[(MPLS Stack) mpls-label-stack]
     |     +-[(MPLS SR Stack) mpls-sr-stack]
     |     +-[(IPv6 SR Stack) srv6-stack]
     |     +-[tunnel-template]
     ...
     |
     +-[tunnel-template]
     |     +-[Extensible: True]
     |     +-[(address) local-address]
     |     +-[(address) remote-address]
     |     +-[mtu]
     |     +-[(Enumeration - ipv4(0), ipv6(1), dual(2) payload_type:]
     |     +-[(Enumeration - ip-in-ip(0),
                   udp(1), gre(2), gtpv1(3), gtpv2(4)) type:]
     |     +-[interface]
     |     +-[next-hop]
     |     +-[gre-key:] (type == gre)
     |     +-[gtp-info] (type == gtpv1 or type == gtpv2 )
     |     |    +-[(Unsigned 32) local-teid]
     |     |    +-[(Unsigned 32) remote-teid]
     |     |    +-[(Boolean) sequence-numbers-on] (type == gtpv1)
```

Figure 32: Tunnel Templates

6.4. Action Templates

The following figure shows common next-hop (set next-hop) and tunnel templates for Actions.

Drop action has no values.

Rewrite uses a Descriptor to set the values of the packet. Exactly one Descriptor MUST be present. Only the Destination and Source port fields, if present, are used from the Descriptor.

Copy-Forward creates a copy of the packet and then forwards it in accordance to the nexthop value.

```
                      |
                      +-[drop-template]
                      ...
                      |
                      +-[rewrite-template]
                      |     +-[Extensible: True]
                      |     +-[ip-prefix-template]
                      |     +-[pmip-traffic-selector]
                      |     +-[rfc5777-classifier]
                      ...
                      |
                      +-[copy-forward-template]
                      |     +-[Extensible: True]
                      |     +-[next-hop:]
```

Figure 33: Action Templates

6.5. Quality of Service Action Templates

PMIP QoS is specified in [RFC7222].

```
       |
       +-[qos-template]
       |     +-[Extensible: True]
       |     +-[(dscp) trafficclass]
       |     +-[pmip-qos]
       |     |     +-[(Unsigned 32) per-mn-agg-max-dl]
       |     |     +-[(Unsigned 32) per-mn-agg-max-ul]
       |     |     +-[per-session-agg-max-dl]
       |     |     |    +-[(Unsigned 32) max-rate:]
       |     |     |    +-[(Boolean) service-flag:]
       |     |     |    +-[(Boolean) exclude-flag:]
       |     |     +-[per-session-agg-max-ul]
       |     |     |    +-[(Unsigned 32) max-rate:]
       |     |     |    +-[(Boolean) service-flag:]
       |     |     |    +-[(Boolean) exclude-flag:]
       |     |     +-[allocation-retention-priority]
       |     |     |    +-[(Unsigned 8) priority-level:]
       |     |     |    +-[(Enumeration) preemption-capability:]
       |     |     |    +-[(Enumeration) preemption-vulnerability:]
       |     |     +-[(Unsigned 32) agg-max-dl]
       |     |     +-[(Unsigned 32) agg-max-ul]
       |     |     +-[(Unsigned 32) gbr-dl]
       |     |     +-[(Unsigned 32) gbr-ul]
```

Figure 34: QoS Templates

6.6. PMIP Command-Set

The following Command Set values are supported for IETF PMIP.

- o assign-ip - Assign the IP Address for the mobile session.
- o assign-dpn - Assign the Data-plane Node.
- o session - Assign values for the Session Level.
- o uplink - Command applies to uplink.
- o downlink - Command applies to downlink.

6.7. 3GPP Specific Templates and Command-Set

3GPP support is optional and detailed in this section. The following acronyms are used:

APN-AMBR:	Access Point Name Aggregate Maximum Bit Rate
UE-AMBR:	User Equipment Aggregate Maximum Bit Rate
QCI:	QoS Class Identifier
EBI:	EPS Bearer Identity
LBI:	Linked Bearer Identity
IMSI:	International Mobile Subscriber Identity
TFT:	Traffic Flow Template (TFT)

Generally, 3GPP QoS values should use the qos-template. Note: User Equipment Aggregate Maximum Bit Rate (UE-AMBR) maps to the per-mn-agg-max-dl and per-mn-agg-max-ul.

```
                      |
                      +-[ MN-Policy-Template ]
                      |     +-[(Unsigned 64) imsi:]
                      ...
                      +-[tunnel-template]
                      |     +-[Extensible: True]
                      |     +-[(unsigned 4) ebi:]
                      |     +-[(unsigned 4) lbi]
                      ...
                      +-[qos-template]
                      |     +-[Extensible: True]
                      |     +-[(unsigned 4) qos-class-identifier]
                      |     +-[(Unsigned 32) ue-agg-max-bitrate]
                      |     +-[(Unsigned 32) apn-agg-max-bitrate]
                      ...
```

Figure 35: 3GPP Mobility Templates

```
            |
            +-[ packet-filter ]
            |     +-[Extensible: True]
            |     +-[(Unsigned 8) identifier:]
            |     +-[Contents:] <List>
            |     |     +-[(ip-address) ipv4-ipv6-local]
            |     |     +-[(ipv6-prefix) ipv6-prefix-local]
            |     |     +-[(ip-address) ipv4-ipv6-remote]
            |     |     +-[(ipv6-prefix) ipv6-prefix-remote]
            |     |     +-[(Unsigned 8) protocol-next-header]
            |     |     +-[(Unsigned 16) local-port]
            |     |     +-[local-port-range]
            |     |     |      +-[(Unsigned 16) local-port-lo]
            |     |     |      +-[(Unsigned 16) local-port-hi]
            |     |     +-[(Unsigned 16) remote-port]
            |     |     +-[remote-port-range]
            |     |     |      +-[(Unsigned 16) remote-port-lo]
            |     |     |      +-[(Unsigned 16) remote-port-hi]
            |     |     +-[(Unsigned 32) sec-parameter-index]
            |     |     +-[(dscp) traffic-class]
            |     |     +-[traffic-class-range]
            |     |     |      +-[(dscp) traffic-class-lo]
            |     |     |      +-[(dscp) traffic-class-hi]
            |     |     +-[(dscp) flow-label]
            ...
```

Figure 36: 3GPP Packet Filter Template (Descriptor)

The following Command Set values are supported for 3GPP.

- o assign-ip - Assign the IP Address for the mobile session.

- o assign-fteid-ip - Assign the Fully Qualified TEID (F-TEID) LOCAL IP address.

- o assign-fteid-teid - Assign the Fully Qualified TEID (F-TEID) LOCAL TEID.

- o session - Assign values for the Session Level. When this involves ’assign-fteid-ip’ and ’assign-fteid-teid’, the values are part of the default bearer.

- o uplink - Command applies to uplink.

- o downlink - Command applies to downlink.

- o assign-dpn - Assign the Data-plane Node.

7. Implementation Status

Three FPC Agent implementations have been made to date. The first was based upon Version 03 of the draft and followed Model 1. The second follows Version 04 of the document. Both implementations were OpenDaylight plug-ins developed in Java by Sprint. Version 04 is now primarily enhanced by GS Labs. Version 03 was known as fpcagent and version 04’s implementation is simply referred to as ’fpc’. A third has been developed on an ONOS Controller for use in MCORD projects.

fpcagent’s intent was to provide a proof of concept for FPC Version 03 Model 1 in January 2016 and research various errors, corrections and optimizations that the Agent could make when supporting multiple DPNs.

As the code developed to support OpenFlow and a proprietary DPN from a 3rd party, several of the advantages of a multi-DPN Agent became obvious including the use of machine learning to reduce the number of Flows and Policy entities placed on the DPN. This work has driven new efforts in the DIME WG, namely Diameter Policy Groups [I-D.bertz-dime-policygroups].

A throughput performance of tens per second using various NetConf based solutions in OpenDaylight made fpcagent, based on version 03, undesirable for call processing. The RPC implementation improved throughput by an order of magnitude but was not useful based upon FPC’s Version 03 design using two information models. During this time the features of version 04 and its converged model became attractive and the fpcagent project was closed in August 2016.

fpcagent will no longer be developed and will remain a proprietary implementation.

The learnings of fpcagent has influenced the second project, fpc. Fpc is also an OpenDaylight project but is an open source release as the Opendaylight FpcAgent plugin (https://wiki.opendaylight.org/view/Project_Proposals:FpcAgent). This project is scoped to be a fully compliant FPC Agent that supports multiple DPNs including those that communicate via OpenFlow. The following features present in this draft and others developed by the FPC development team have already led to an order of magnitude improvement.

Migration of non-realtime provisioning of entities such as topology and policy allowed the implementation to focus only on the rpc.

Using only 5 messages and 2 notifications has also reduced implementation time.

Command Sets, an optional feature in this specification, have eliminated 80% of the time spent determining what needs to be done with a Context during a Create or Update operation.

Op Reference is an optional feature modeled after video delivery. It has reduced unnecessary cache lookups. It also has the additional benefit of allowing an Agent to become cacheless and effectively act as a FPC protocol adapter remotely with multi-DPN support or co-located on the DPN in a single-DPN support model.

Multi-tenant support allows for Cache searches to be partitioned for clustering and performance improvements. This has not been capitalized upon by the current implementation but is part of the development roadmap.

Use of Contexts to pre-provision policy has also eliminated any processing of Ports for DPNs which permitted the code for CONFIGURE and CONF_BUNDLE to be implemented as a simple nested FOR loops (see below).

Initial v04 performance results without code optimizations or tuning allow reliable provisioning of 1K FPC Mobility-Contexts processed per second on a 12 core server. This results in 2x the number of transactions on the southbound interface to a proprietary DPN API on the same machine.

fpc currently supports the following:

1 proprietary DPN API

```
                                       Policy and Topology as defined in this
                                       specification using OpenDaylight North Bound
                                       Interfaces such as NetConf and RestConf

                                       CONFIG and CONF_BUNDLE (all operations)

                                       DPN assignment, Tunnel allocations and IPv4
                                       address assignment by the Agent or Client.

                                       Immediate Response is always an
                                       OK_NOTIFY_FOLLOWS.
```

```
        assignment system (receives rpc call):
          perform basic operation integrity check
          if CONFIG then
            goto assignments
            if assignments was ok then
              send request to activation system
              respond back to client with assignment data
            else
              send back error
            end if
          else if CONF_BUNDLE then
            for each operation in bundles
            goto assignments
            if assignments was ok then
              hold onto data
            else
              return error with the assignments that occurred in
                prior operations (best effort)
            end if
            end for
            send bundles to activation systems
          end if

        assignments:
          assign DPN, IPv4 Address and/or tunnel info as required
          if an error occurs undo all assignments in this operation
          return result

        activation system:
          build cache according to op-ref and operation type
          for each operation
            for each Context
              for each DPN / direction in Context
                perform actions on DPN according to Command Set
              end for
            end for
          end for
          commit changes to in memory cache
          log transaction for tracking and notification
                                          (CONFIG_RESULT_NOTIFY)
```

Figure 37: fpc pseudo code

For further information please contact Lyle Bertz who is also a co-author of this document.

NOTE: Tenant support requires binding a Client ID to a Tenant ID (it is a one to many relation) but that is outside of the scope of this specification. Otherwise, the specification is complete in terms of providing sufficient information to implement an Agent.

8. Security Considerations

Detailed protocol implementations for DMM Forwarding Policy Configuration must ensure integrity of the information exchanged between a FPC Client and a FPC Agent. Required Security Associations may be derived from co-located functions, which utilize the FPC Client and FPC Agent respectively.

The YANG modules defined in this memo are designed to be accessed via the NETCONF [RFC6241] or RESTCONF [RFC8040] protocol. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

The information model defined in the memo is designed to be access by protocols specified in extensions to this document or, if using the YANG modules, as described above.

There are a number of data nodes defined which are writable/creatable/deletable. These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., a NETCONF edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

> Nodes under the Policy tree provide generic policy enforcement and traffic classification. They can be used to block or permit traffic. If this portion of the model was to be compromised it may be used to block, identify or permit traffic that was not intended by the Tenant or FPC Client.
>
> Nodes under the Topology tree provide definition of the Tenant’s forwarding topology. Any compromise of this information will provide topology information that could be used for subsequent attack vectors. Removal of topology can limit services.
>
> Mobility-Context provides runtime only information and manipulated by remote procedure calls. The unwanted deletion or removal of such information would deny users service or provide services to unauthorized parties.

Some of the readable data nodes defined may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

IP address assignments in the Mobility-Context along with their associated tunnel configurations/identifiers (from the FPC base module)

Internaitonal Mobile Subscriber Identity (IMSI) and bearer identifiers in the Context when using the FPC base model

Some of the RPC operations defined may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

Configure sends Mobility-Context information which can include information of a sensitive or vulnerable nature in some network environments as described above.

Monitor related RPC operations do not specifically provide sensitive or vulnerable information but care must be taken by users to avoid identifier values that expose sensitive or vulnerable information.

Notifications MUST be treated with same level of protection and scrutiny as the operations they correspond to. For example, a Configure-Result-Notification provides the same information that is sent as part of the input and output of the Configure RPC operation.

General usage of FPC MUST consider the following:

FPC Naming Section 4.5 permits arbitrary string values but a user MUST avoid placing sensitive or vulnerable information in those values.

Policies that are very narrow and permit the identification of specific traffic, e.g. that of a single user, SHOULD be avoided.

9. IANA Considerations

This document registers six URIs in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations have been made.

URI: urn:ietf:params:xml:ns:yang:ietf-dmm-fpc
Registrant Contact: The DMM WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

```
       URI: urn:ietf:params:xml:ns:yang:ietf-dmm-pmip-qos
       Registrant Contact: The DMM WG of the IETF.
       XML: N/A, the requested URI is an XML namespace.

       URI: urn:ietf:params:xml:ns:yang:ietf-dmm-traffic-selector-types
       Registrant Contact: The DMM WG of the IETF.
       XML: N/A, the requested URI is an XML namespace.

       URI: urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-settingsext
       Registrant Contact: The DMM WG of the IETF.
       XML: N/A, the requested URI is an XML namespace.

      URI: urn:ietf:params:xml:ns:yang:ietf-diam-trafficclassifier
      Registrant Contact: The DMM WG of the IETF.
      XML: N/A, the requested URI is an XML namespace.
```

This document registers the following YANG modules in the "YANG Module Names" registry [RFC6020].

```
         name:         ietf-dmm-fpc
         namespace:    urn:ietf:params:xml:ns:yang:ietf-dmm-fpc
         prefix:       fpc
         reference:    TBD1

         name:         ietf-dmm-pmip-qos
         namespace:    urn:ietf:params:xml:ns:yang:ietf-dmm-pmip-qos
         prefix:       qos-pmip
         reference:    TBD2

         name:         ietf-dmm-traffic-selector-types
         namespace:    urn:ietf:params:xml:ns:yang:
           ietf-dmm-traffic-selector-types
         prefix:       traffic-selectors
         reference:    TBD3

         name:         ietf-dmm-fpc-settingsext
         namespace:    urn:ietf:params:xml:ns:yang:
           ietf-dmm-fpc-settingsext
         prefix:       fpcbase
         reference:    TBD4

           name:         ietf-diam-trafficclassifier
           namespace:    urn:ietf:params:xml:ns:yang:
           ietf-diam-trafficclassifier
           prefix:       diamclassifier
           reference:    TBD5
```

10. Work Team Participants

Participants in the FPSM work team discussion include Satoru Matsushima, Danny Moses, Sri Gundavelli, Marco Liebsch, Pierrick Seite, Alper Yegin, Carlos Bernardos, Charles Perkins and Fred Templin.

11. References

11.1. Normative References

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-13 (work in progress), May 2018.

[I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-14 (work in progress), June 2018.

[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>.

[RFC5777] Korhonen, J., Tschofenig, H., Arumaithurai, M., Jones, M., Ed., and A. Lior, "Traffic Classification and Quality of Service (QoS) Attributes for Diameter", RFC 5777, DOI 10.17487/RFC5777, February 2010, <https://www.rfc-editor.org/info/rfc5777>.

[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>.

[RFC6088] Tsirtsis, G., Giarreta, G., Soliman, H., and N. Montavont, "Traffic Selectors for Flow Bindings", RFC 6088, DOI 10.17487/RFC6088, January 2011, <https://www.rfc-editor.org/info/rfc6088>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8072] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", RFC 8072, DOI 10.17487/RFC8072, February 2017, <https://www.rfc-editor.org/info/rfc8072>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

11.2. Informative References

[I-D.bertz-dime-policygroups]
Bertz, L. and M. Bales, "Diameter Policy Groups and Sets", draft-bertz-dime-policygroups-05 (work in progress), December 2017.

[I-D.ietf-dmm-deployment-models]
Gundavelli, S. and S. Jeon, "DMM Deployment Models and Architectural Considerations", draft-ietf-dmm-deployment-models-04 (work in progress), May 2018.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, DOI 10.17487/RFC3958, January 2005, <https://www.rfc-editor.org/info/rfc3958>.

[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>.

[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>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7222] Liebsch, M., Seite, P., Yokota, H., Korhonen, J., and S. Gundavelli, "Quality-of-Service Option for Proxy Mobile IPv6", RFC 7222, DOI 10.17487/RFC7222, May 2014, <https://www.rfc-editor.org/info/rfc7222>.

[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>.

[RFC7660] Bertz, L., Manning, S., and B. Hirschman, "Diameter Congestion and Filter Attributes", RFC 7660, DOI 10.17487/RFC7660, October 2015, <https://www.rfc-editor.org/info/rfc7660>.

Appendix A. YANG Data Model for the FPC protocol

This section provides a type mapping for FPC structures in YANG. When being mapped to a specific information such as YANG the data type MAY change.

Keys for Actions, Descriptors, Rules, Policies, DPNs, Domains and Mobility-Contexts are specified as FPC-Identity which follows rules according to Section 4.5.

Action and Descriptor Templates are mapped as choices. This was done to ensure no duplication of Types and avoid use of identityref for typing.

Policy Expressions are provided as default values. NOTE that a static value CANNOT be supported in YANG.

Mapping of templates to YANG are performed as follows:

Value is defined as a choice statement for extensibility and therefore a type value is not necessary to discriminated types

Generic attributes are distinguished by the "Settings" type and holds ANY value. It is an any data node under configurations.

The CONFIGURE and CONFIGURE-RESULT-NOTIFICATION use the yang-patch-status which is a container for edits. This was done to maximize YANG reuse.

In the configure rpc, operation-id is mapped to patch-id and in an edit the edit-type is mapped to operation.

The Result-Status attribute is mapped to the ’ok’ (empty leaf) or errors structure.

The Policy-Status is mapped to entity-state to reduce YANG size.

Five modules are defined:

o ietf-dmm-fpc (fpc) - Defines the base model and messages for FPC that are meant to be static in FPC.

o ietf-dmm-fpc-settingsext - A FPC module that defines the information model elements that are likely to be extended in FPC.

o ietf-pmip-qos (pmip-qos) - Defines proxy mobile IPv6 QoS parameters per RFC 7222

o ietf-trafficselectors-types (traffic-selectors) - Defines Traffic Selectors per [RFC6088]

o ietf-diam-trafficclassifier (diamclassifier) - Defines the Classifier per [RFC5777]

All modules defined in this specification make use of (import) ietf-inet-types as defined in [RFC6991].

ietf-dmm-fpc-settingsext and ietf-diam-trafficclassifier make use of (imports) ietf-yang-types as defined in [RFC6991].

ietf-dmm-fpc imports the restconf (ietf-restconf) [RFC8040] and yang patch (ietf-yang-patch) [RFC8072] modules.

ietf-pmip-qos and ietf-dmm-fpc-settings import the trafficselector from the ietf-traffic-selector-types module.

ietf-dmm-fpc-settings also imports the qosattribute (ietf-pmip-qos) and classifier (ietf-diam-trafficclassifier).

ietf-dmm-fpc-settingsext groups various settings, actions and descriptors and is used by the fpc module (ietf-dmm-fpc).

The following groupings are intended for reuse (import) by other modules.

o qosoption (ietf-qos-pmip module)

- o qosattribute (ietf-qos-pmip module)
- o qosoption (ietf-qos-pmip module)
- o Allocation-Retention-Priority-Value (ietf-qos-pmip module)
- o trafficselector (ietf-traffic-selector-types)
- o classifier (ietf-diam-trafficclassifier)
- o packet-filter (ietf-dmm-fpc-settingsext)
- o instructions (ietf-dmm-fpc-settingsext)
- o fpc-descriptor-value (ietf-dmm-fpc-settingsext)
- o fpc-action-value (ietf-dmm-fpc-settingsext)

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

DPNs conformant to NMDA MAY only have policies, installed policies, topology, domains and mobility session information that has been assigned to it in its intended and operational datastores. What is housed in the operational datastore MAY be determined on a per DPN basis and using the Entity-Status as a guideline based upon tradeoffs described in Section 4.6.

ServiceGroups are not expected to appear in operational datastores of DPNs as they remain in and are used by FPC Agents and Clients. They MAY be operationally present in DNS when using the Dynamic Delegation and Discovery System (DDDS) as defined in [RFC3958] or the operational datastore of systems that provide equivalent functionality.

A.1. FPC YANG Model

This module defines the information model and protocol elements specified in this document.

This module references [RFC6991], [RFC8040] and the fpc-settingsext module defined in this document.

```
<CODE BEGINS> file "ietf-dmm-fpc@2018-05-17.yang"
module ietf-dmm-fpc {
  yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc";
    prefix fpc;
```

```
       import ietf-inet-types { prefix inet;
           revision-date 2013-07-15; }
       import ietf-dmm-fpc-settingsext { prefix fpcbase;
           revision-date 2018-05-17; }
       import ietf-diam-trafficclassifier { prefix rfc5777;
           revision-date 2018-05-17; }
       import ietf-restconf { prefix rc;
           revision-date 2017-01-26; }
       import ietf-yang-patch { prefix ypatch;
           revision-date 2017-02-22; }

       organization "IETF Distributed Mobility Management (DMM)
         Working Group";

       contact
          "WG Web:   <http://tools.ietf.org/wg/netmod/>
           WG List:  <mailto:netmod@ietf.org>

           WG Chair: Dapeng Liu
                     <mailto:maxpassion@gmail.com>

           WG Chair: Jouni Korhonen
                     <mailto:jouni.nospam@gmail.com>

           Editor:   Satoru Matsushima
                     <mailto:satoru.matsushima@g.softbank.co.jp>

           Editor:   Lyle Bertz
                     <mailto:lylebe551144@gmail.com>";

       description
       "This module contains YANG definition for
        Forwarding Policy Configuration Protocol (FPCP).

        Copyright (c) 2016 IETF Trust and the persons identified as the
        document authors. All rights reserved.

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

       revision 2018-05-17 {
         description "Initial Revision.";
         reference "draft-ietf-dmm-fpc-cpdp-10";
       }

         //General Structures
         grouping templatedef {
             leaf extensible {
                 type boolean;
                 description "Indicates if the template is extensible";
             }
             leaf-list static-attributes {
                 type string;
                 description "Attribute (Name) whose value cannot
                     change";
             }
             leaf-list mandatory-attributes {
                 type string;
                 description "Attribute (Name) of optional attributes
                   that MUST be present in instances of this tempplate.";
             }
             leaf entity-state {
                 type enumeration {
                     enum initial {
                         description "Inital Configuration";
                     }
                     enum partially-configured {
                         description "Partial Configuration";
                     }
                     enum configured {
                         description "Confgured";
                     }
                     enum active {
                         description "Active";
                     }
                 }
                 default initial;
                 description "Entity State";
             }
             leaf version {
                 type uint32;
                 description "Template Version";
             }
             description "Teamplate Definition";
         }
         typedef fpc-identity {
             type union {
                 type uint32;
                 type instance-identifier;
```

```
                type string;
            }
            description "FPC Identity";
        }
        grouping index {
            leaf index {
                type uint16;
                description "Index";
            }
            description "Index Value";
        }

        // Policy Structures
        grouping descriptor-template-key {
            leaf descriptor-template-key {
                type fpc:fpc-identity;
                mandatory true;
                description "Descriptor Key";
            }
            description "Descriptor-Template Key";
        }
        grouping action-template-key {
            leaf action-template-key {
                type fpc:fpc-identity;
                mandatory true;
                description "Action Key";
            }
            description "Action-Template Key";
        }
        grouping rule-template-key {
            leaf rule-template-key {
                type fpc:fpc-identity;
                mandatory true;
                description "Rule Identifier";
            }
            description "Rule Key";
        }
        grouping policy-template-key {
            leaf policy-template-key {
                type fpc:fpc-identity;
                mandatory true;
                description "Rule Identifier";
            }
            description "Rule Key";
        }

        grouping fpc-setting-value {
            anydata setting;
            description "FPC Setting Value";
        }
        // Configuration / Settings
        grouping policy-configuration-choice {
               choice policy-configuration-value {
                    case descriptor-value {
                        uses fpcbase:fpc-descriptor-value;
                        description "Descriptor Value";
                    }
                    case action-value {
                        uses fpcbase:fpc-action-value;
                        description "Action Value";
                    }
                    case setting-value {
                        uses fpc:fpc-setting-value;
                        description "Setting";
                    }
                    description "Policy Attributes";
                }
                description "Policy Configuration Value Choice";
        }
        grouping policy-configuration {
            list policy-configuration {
                key index;
                uses fpc:index;
                uses fpc:policy-configuration-choice;
                description "Policy Configuration";
            }
            description "Policy Configuration Value";
        }
        grouping ref-configuration {
            uses fpc:policy-template-key;
            uses fpc:policy-configuration;
            uses fpc:templatedef;
            description "Policy-Configuration Entry";
        }

        // FPC Policy
        grouping policy-information-model {
          list action-template {
            key action-template-key;
            uses fpc:action-template-key;
            uses fpcbase:fpc-action-value;
            uses fpc:templatedef;
            description "Action Template";
          }
          list descriptor-template {
            key descriptor-template-key;
```

```
             uses fpc:descriptor-template-key;
             uses fpcbase:fpc-descriptor-value;
             uses fpc:templatedef;
             description "Descriptor Template";
           }
           list rule-template {
             key rule-template-key;
             uses fpc:rule-template-key;
             leaf descriptor-match-type {
                 type enumeration {
                     enum or {
                         value 0;
                         description "OR logic";
                     }
                     enum and {
                         value 1;
                         description "AND logic";
                     }
                 }
                 mandatory true;
                 description "Type of Match (OR or AND) applied
                     to the descriptor-configurations";
             }
             list descriptor-configuration {
                 key "descriptor-template-key";
                 uses fpc:descriptor-template-key;
                 leaf direction {
                     type rfc5777:direction-type;
                     description "Direction";
                 }
                 list attribute-expression {
                     key index;
                     uses fpc:index;
                     uses fpcbase:fpc-descriptor-value;
                     description "Descriptor Attributes";
                 }
                 uses fpc:fpc-setting-value;
                 description "A set of Descriptor references";
             }
             list action-configuration {
                 key "action-order";
                 leaf action-order {
                     type uint32;
                     mandatory true;
                     description "Action Execution Order";
                 }
                 uses fpc:action-template-key;
                 list attribute-expression {
                     key index;
                     uses fpc:index;
                     uses fpcbase:fpc-action-value;
                     description "Action Attributes";
                 }
                 uses fpc:fpc-setting-value;
                 description "A set of Action references";
             }
             uses fpc:templatedef;
             list rule-configuration {
                 key index;
                 uses fpc:index;
                 uses fpc:policy-configuration-choice;
                 description "Rule Configuration";
             }
             description "Rule Template";
           }
           list policy-template {
             key policy-template-key;
             uses fpc:policy-template-key;
             list rule-template {
                 key "precedence";
                 unique "rule-template-key";
                 leaf precedence {
                     type uint32;
                     mandatory true;
                     description "Rule Precedence";
                 }
                 uses fpc:rule-template-key;
                 description "Rule Entry";
             }
             uses fpc:templatedef;
             uses fpc:policy-configuration;
             description "Policy Template";
           }
           description "FPC Policy Structures";
         }

         // Topology Information Model
         identity role {
             description "Role";
         }
         grouping dpn-key {
             leaf dpn-key {
                 type fpc:fpc-identity;
                 description "DPN Key";
             }
             description "DPN Key";
         }
         grouping role-key {
             leaf role-key {
                 type identityref {
                     base "fpc:role";
                 }
                 mandatory true;
                 description "Access Technology Role";
             }
             description "Access Technology Role key";
         }
         grouping interface-key {
             leaf interface-key{
                 type fpc:fpc-identity;
                 mandatory true;
                 description "interface identifier";
             }
             description "Interface Identifier key";
         }
         identity interface-protocols {
             description "Protocol supported by the interface";
         }
         identity features {
             description "Protocol features";
         }

       // Mobility Context
       grouping mobility-context {
         leaf mobility-context-key {
           type fpc:fpc-identity;
           mandatory true;
           description "Mobility Context Key";
         }
         leaf-list delegating-ip-prefix {
             type inet:ip-prefix;
             description "IP Prefix";
         }
         leaf parent-context {
           type fpc:fpc-identity;
           description "Parent Mobility Context";
         }
         leaf-list child-context {
             type fpc:fpc-identity;
             description "Child Mobility Context";
         }
         container mobile-node {
             leaf-list ip-address {
                 type inet:ip-address;
```

```
                description "IP Address";
            }
            leaf imsi {
                type fpcbase:imsi-type;
                description "IMSI";
            }
            list mn-policy-configuration {
                key policy-template-key;
                uses fpc:ref-configuration;
                description "MN Policy Configuration";
            }
            description "Mobile Node";
        }
        container domain {
            leaf domain-key {
                type fpc:fpc-identity;
                description "Domain Key";
            }
            list domain-policy-settings {
                key policy-template-key;
                uses fpc:ref-configuration;
                description "MN Policy Configuration";
            }
            description "Domain";
        }
        list dpn {
            key dpn-key;
            uses fpc:dpn-key;
            list dpn-policy-configuration {
                key policy-template-key;
                uses fpc:ref-configuration;
                description "DPN Policy Configuration";
            }
            leaf role {
                type identityref {
                    base "fpc:role";
                }
                description "Role";
            }
            list service-data-flow {
                key identifier;
                leaf identifier {
                    type uint32;
                    description "Generic Identifier";
                }
                leaf service-group-key {
                    type fpc:fpc-identity;
                    description "Service Group Key";
```

```
               }
               list interface {
                   key interface-key;
                   uses fpc:interface-key;
                   description "interface assigned";
               }
               list service-data-flow-policy-configuration {
                   key policy-template-key;
                   uses fpc:ref-configuration;
                   description "Flow Policy Configuration";
               }
               description "Service Dataflow";
           }
           description "DPN";
       }
       description "Mobility Context";
     }

     // Events, Probes & Notifications
     identity event-type {
       description "Base Event Type";
     }
     typedef event-type-id {
       type uint32;
       description "Event ID Type";
     }
     grouping monitor-key {
       leaf monitor-key {
         type fpc:fpc-identity;
         mandatory true;
         description "Monitor Key";
       }
       description "Monitor Id";
     }
     grouping monitor-config {
       uses fpc:templatedef;
       uses fpc:monitor-key;
       leaf target {
         type string;
         description "target";
       }
       leaf deferrable {
         type boolean;
         description "Indicates reports related to this
           config can be delayed.";
       }
       choice configuration {
         mandatory true;
         leaf period {
             type uint32;
             description "Period";
         }
         case threshold-config {
           leaf low {
             type uint32;
             description "low threshold";
           }
           leaf hi {
             type uint32;
             description "high threshold";
           }
           description "Threshold Config Case";
         }
         leaf schedule {
             type uint32;
             description "Reporting Time";
         }
         leaf-list event-identities {
             type identityref {
               base "fpc:event-type";
             }
             description "Event Identities";
         }
         leaf-list event-ids {
             type uint32;
               description "Event IDs";
         }
         description "Event Config Value";
       }
       description "Monitor Configuration";
     }

     // Top Level Structures
     list tenant {
      key "tenant-key";
      leaf tenant-key {
           type fpc:fpc-identity;
           description "Tenant Key";
      }
      container topology-information-model {
        config false;
        list service-group {
           key "service-group-key role-key";
           leaf service-group-key {
               type fpc:fpc-identity;
               mandatory true;
```

```
                description "Service Group Key";
            }
            leaf service-group-name {
                type string;
                description "Service Group Name";
            }
            uses fpc:role-key;
            leaf role-name {
                type string;
                mandatory true;
                description "Role Name";
            }
            leaf-list protocol {
              type identityref {
                base "interface-protocols";
              }
              min-elements 1;
              description "Supported protocols";
            }
            leaf-list feature {
                type identityref {
                    base "interface-protocols";
                }
                description "Supported features";
            }
            list service-group-configuration {
                key index;
                uses fpc:index;
                uses fpc:policy-configuration-choice;
                description "Settings";
            }
            list dpn {
                key dpn-key;
                uses fpc:dpn-key;
                min-elements 1;
                list referenced-interface {
                    key interface-key;
                    uses fpc:interface-key;
                    leaf-list peer-service-group-key {
                        type fpc:fpc-identity;
                        description "Peer Service Group";
                    }
                    description "Referenced Interface";
                }
                description "DPN";
            }
            description "Service Group";
        }
```

```
       list dpn {
         key dpn-key;
         uses fpc:dpn-key;
         leaf dpn-name {
           type string;
           description "DPN name";
         }
         leaf dpn-resource-mapping-reference {
           type string;
           description "Reference to underlying DPN resource(s)";
         }
         leaf domain-key {
           type fpc:fpc-identity;
           description "Domains";
         }
         leaf-list service-group-key {
           type fpc:fpc-identity;
           description "Service Group";
         }
         list interface {
           key "interface-key";
           uses fpc:interface-key;
           leaf interface-name {
               type string;
               description "Service Endpoint Interface Name";
           }
           leaf role {
               type identityref {
                   base "fpc:role";
               }
               description "Roles supported";
           }
           leaf-list protocol {
             type identityref {
               base "interface-protocols";
             }
             description "Supported protocols";
           }
           list interface-configuration {
            key index;
            uses fpc:index;
            uses fpc:policy-configuration-choice;
            description "Interface settings";
          }
          description "DPN interfaces";
         }
         list dpn-policy-configuration {
           key policy-template-key;
           uses fpc:ref-configuration;
           description "DPN Policy Configuration";
         }
         description "Set of DPNs";
       }
       list domain {
         key domain-key;
         leaf domain-key {
           type fpc:fpc-identity;
           mandatory true;
           description "Domain Key";
         }
         leaf domain-name {
           type string;
           description "Domain displayname";
         }
         list domain-policy-configuration {
           key policy-template-key;
           uses fpc:ref-configuration;
           description "Domain Configuration";
         }
         description "List of Domains";
       }
       container dpn-checkpoint {
           uses fpc:basename-info;
           description "DPN Checkpoint information";
       }
       container service-group-checkpoint {
           uses fpc:basename-info;
           description "Service Group Checkpoint information";
       }
       container domain-checkpoint {
           uses fpc:basename-info;
           description "Domain Checkpoint information";
       }
       description "FPC Topology grouping";
      }
      container policy-information-model {
           config false;
           uses fpc:policy-information-model;
           uses fpc:basename-info;
           description "Policy";
      }
      list mobility-context {
           key "mobility-context-key";
           config false;
           uses fpc:mobility-context;
           description "Mobility Context";
      }
      list monitor {
           key monitor-key;
           config false;
           uses fpc:monitor-config;
           description "Monitor";
      }
      description "Tenant";
     }

       typedef agent-identifier {
           type fpc:fpc-identity;
           description "Agent Identifier";
       }
       typedef client-identifier {
           type fpc:fpc-identity;
           description "Client Identifier";
       }
       grouping basename-info {
             leaf basename {
               type fpc:fpc-identity;
               description "Rules Basename";
             }
             leaf base-checkpoint {
               type string;
               description "Checkpoint";
             }
             description "Basename Information";
       }

       // RPCs
       grouping client-id {
           leaf client-id {
               type fpc:client-identifier;
               mandatory true;
               description "Client Id";
           }
           description "Client Identifier";
       }
       grouping execution-delay {
           leaf execution-delay {
               type uint32;
               description "Execution Delay (ms)";
           }
           description "Execution Delay";
       }
       typedef ref-scope {
         type enumeration {
           enum none {
             value 0;
             description "no references";
           }
           enum op {
             value 1;
             description "All references are intra-operation";
           }
           enum bundle {
             value 2;
             description "All references in exist in bundle";
           }
           enum storage {
             value 3;
             description "One or more references exist in storage.";
           }
           enum unknown {
             value 4;
             description "The location of the references are unknown.";
           }
         }
         description "Search scope for references in the operation.";
       }
       rpc configure {
           description "Configure RPC";
           input {
               uses client-id;
               uses execution-delay;
               uses ypatch:yang-patch;
           }
           output {
               uses ypatch:yang-patch-status;
           }
       }
       augment "/configure/input/yang-patch/edit" {
           leaf reference-scope {
               type fpc:ref-scope;
               description "Reference Scope";
           }
           uses fpcbase:instructions;
           description "yang-patch edit augments for configure rpc";
       }
       grouping subsequent-edits {
            list subsequent-edit {
              key edit-id;
              ordered-by user;

              description "Edit list";
              leaf edit-id {
                type string;
                description "Arbitrary string index for the edit.";
              }

              leaf operation {
                type enumeration {
                  enum create {
                    description "Create";
                  }
                  enum delete {
                    description "Delete";
                  }
                  enum insert {
                    description "Insert";
                  }
                  enum merge {
                    description "Merge";
                  }
                  enum move {
                    description "Move";
                  }
                  enum replace {
                    description "Replace";
                  }
                  enum remove {
                    description
                      "Delete the target node if it currently exists.";
                  }
                }
                mandatory true;
                description
                  "The datastore operation requested";
              }

              leaf target {
                type ypatch:target-resource-offset;
                mandatory true;
                description
                  "Identifies the target data node";
              }

              leaf point {
               when "(../operation = 'insert' or ../operation = 'move')"
                + "and (../where = 'before' or ../where = 'after')" {
                  description
                    "This leaf only applies for 'insert' or 'move'
                     operations, before or after an existing entry.";
                }
                type ypatch:target-resource-offset;
                description
                  "The absolute URL path for the data node";
              }

              leaf where {
               when "../operation = 'insert' or ../operation = 'move'" {
                  description
                    "This leaf only applies for 'insert' or 'move'
                     operations.";
                }
                type enumeration {
                  enum before {
                    description
                      "Insert or move a data node before.";
                  }
                  enum after {
                    description
                      "Insert or move a data node after.";
                  }
                 enum first {
                    description
                      "Insert or move a data node so it becomes ordered
                       as the first entry.";
                  }
                  enum last {
                    description
                      "Insert or move a data node so it becomes ordered
                       as the last entry.";
                  }
                }
                default last;
                description
                  "Identifies where a data resource will be inserted
                   or moved.";
              }

              anydata value {
                when "../operation = 'create' "
                   + "or ../operation = 'merge' "
                   + "or ../operation = 'replace' "
                   + "or ../operation = 'insert'" {
                  description
                    "The anydata 'value' is only used for 'create',
                     'merge', 'replace', and 'insert' operations.";
                }
                description
```

```
                  "Value used for this edit operation.";
              }
            }
            description "Subsequent Edits";
       }
       augment "/configure/output/yang-patch-status/edit-status/edit/"
           + "edit-status-choice/ok" {
           leaf notify-follows {
               type boolean;
               description "Notify Follows Indication";
           }
           uses fpc:subsequent-edits;
           description "Configure output augments";
       }

       grouping op-header {
           uses client-id;
           uses execution-delay;
           leaf operation-id {
               type uint64;
               mandatory true;
               description "Operation Identifier";
           }
           description "Common Operation header";
       }
       grouping monitor-response {
           leaf operation-id {
               type uint64;
               mandatory true;
               description "Operation Identifier";
           }
           choice edit-status-choice {
                  description
                    "A choice between different types of status
                     responses for each 'edit' entry.";
                  leaf ok {
                    type empty;
                    description
                      "This 'edit' entry was invoked without any
                       errors detected by the server associated
                       with this edit.";
                  }
                  case errors {
                    uses rc:errors;
                    description
                      "The server detected errors associated with the
                       edit identified by the same 'edit-id' value.";
                  }
           }
           description "Monitor Response";
       }

     // Common RPCs
     rpc register_monitor {
       description "Used to register monitoring of parameters/events";
       input {
         uses fpc:op-header;
         list monitor {
           key monitor-key;
           uses fpc:monitor-config;
           description "Monitor Configuration";
         }
       }
       output {
           uses fpc:monitor-response;
       }
     }
     rpc deregister_monitor {
       description "Used to de-register monitoring of
         parameters/events";
       input {
         uses fpc:op-header;
         list monitor {
           key monitor-key;
           uses fpc:monitor-key;
           min-elements 1;
           leaf send_data {
               type boolean;
               description "Indicates if NOTIFY with final data
                   is desired upon deregistration";
           }
           description "Monitor Identifier";
         }
       }
       output {
           uses fpc:monitor-response;
       }
     }
     rpc probe {
       description "Probe the status of a registered monitor";
       input {
         uses fpc:op-header;
         list monitor {
           key monitor-key;
           uses fpc:monitor-key;
           min-elements 1;
           description "Monitor";
         }
       }
       output {
           uses fpc:monitor-response;
       }
     }

     // Notification Messages & Structures
     notification config-result-notification {
       uses ypatch:yang-patch-status;
       description "Configuration Result Notification";
     }
     augment "/config-result-notification" {
           uses fpc:subsequent-edits;
           description "config-result-notificatio augment";
     }

     identity notification-cause {
       description "Notification Cause";
     }
     identity subscribed-event-occurred {
       base "notification-cause";
       description "Subscribed Event Occurence";
     }
     identity low-threshold-crossed {
       base "notification-cause";
       description "Subscribed Event Occurence";
     }
     identity high-threshold-crossed {
       base "notification-cause";
       description "Subscribed Event Occurence";
     }
     identity periodic-report {
       base "notification-cause";
       description "Periodic Report";
     }
     identity scheduled-report {
       base "notification-cause";
       description "Scheduled Report";
     }
     identity probe {
       base "notification-cause";
       description "Probe";
     }
     identity deregistration-final-value {
       base "notification-cause";
       description "Probe";
     }
     identity monitoring-suspension {
       base "notification-cause";
       description "Indicates monitoring suspension";
     }
     identity monitoring-resumption {
       base "notification-cause";
       description "Indicates that monitoring has resumed";
     }
     identity dpn-available {
       base "notification-cause";
       description "DPN Candidate Available";
     }
     identity dpn-unavailable {
       base "notification-cause";
       description "DPN Unavailable";
     }
     notification notify {
       leaf notification-id {
         type uint32;
         description "Notification Identifier";
       }
       leaf timestamp {
         type uint32;
         description "timestamp";
       }
       list report {
           key monitor-key;
           uses fpc:monitor-key;
           min-elements 1;
           leaf trigger {
               type identityref {
                   base "notification-cause";
               }
               description "Notification Cause";
           }
           choice value {
               case dpn-candidate-available {
                   leaf node-id {
                       type inet:uri;
                       description "Topology URI";
                   }
                   list supported-interface-list {
                       key role-key;
                       uses fpc:role-key;
                       description "Support Intefaces";
                   }
                   description "DPN Candidate Information";
               }
               case dpn-unavailable {
                   leaf dpn-id {
                     type fpc:fpc-identity;
                     description "DPN Identifier for DPN Unavailable";
                   }
                   description "DPN Unavailable";
               }
               anydata report-value {
                   description "Any non integer report";
               }
               description "Report Value";
           }
           description "Report";
       }
       description "Notify Message";
     }
   }
   <CODE ENDS>
```

A.2. FPC YANG Settings and Extensions Model

This module defines the base data elements in FPC that are likely to be extended.

This module references [RFC6991], ietf-trafficselector-types and ietf-pmip-qos modules.

```
   <CODE BEGINS> file "ietf-dmm-fpc-settingsext@2018-05-17.yang"
   module ietf-dmm-fpc-settingsext {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-settingsext";
     prefix fpcbase;

       import ietf-inet-types { prefix inet;
           revision-date 2013-07-15; }
       import ietf-trafficselector-types { prefix traffic-selectors;
           revision-date 2018-05-17; }
       import ietf-yang-types { prefix ytypes;
           revision-date 2013-07-15; }
       import ietf-pmip-qos { prefix pmipqos;
           revision-date 2018-05-17; }
       import ietf-diam-trafficclassifier { prefix rfc5777;
           revision-date 2018-05-17; }

       organization "IETF Distributed Mobility Management (DMM)
         Working Group";
       contact
          "WG Web:   <http://tools.ietf.org/wg/netmod/>
           WG List:  <mailto:netmod@ietf.org>

           WG Chair: Dapeng Liu
                     <mailto:maxpassion@gmail.com>

           WG Chair: Sri Gundavelli
                     <mailto:sgundave@cisco.com>

           Editor:   Satoru Matsushima
                     <mailto:satoru.matsushima@g.softbank.co.jp>

           Editor:   Lyle Bertz
                     <mailto:lylebe551144@gmail.com>";

       description
       "This module contains YANG definition for
        Forwarding Policy Configuration Protocol(FPCP).

         It contains Settings defintions as well as Descriptor and
         Action extensions.

        Copyright (c) 2016 IETF Trust and the persons identified as the
        document authors. All rights reserved.

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

   revision 2018-05-17 {
       description "Initial Revision.";
       reference "draft-ietf-dmm-fpc-cpdp-10";
   }

       //Tunnel Information
        identity tunnel-type {
             description "Tunnel Type";
         }
         identity grev1 {
             base "fpcbase:tunnel-type";
             description "GRE v1";
         }
         identity grev2 {
             base "fpcbase:tunnel-type";
             description "GRE v2";
         }
         identity ipinip {
             base "fpcbase:tunnel-type";
             description "IP in IP";
         }
         identity gtpv1 {
             base "fpcbase:tunnel-type";
             description "GTP version 1 Tunnel";
         }
         identity gtpv2 {
             base "fpcbase:tunnel-type";
             description "GTP version 2 Tunnel";
         }

       grouping tunnel-value {
         container tunnel-info {
           leaf tunnel-local-address {
               type inet:ip-address;
               description "local tunnel address";
           }
           leaf tunnel-remote-address {
               type inet:ip-address;
               description "remote tunnel address";
           }
           leaf mtu-size {
               type uint32;
               description "MTU size";
           }
           leaf tunnel {
               type identityref {
                   base "fpcbase:tunnel-type";
               }
           description "tunnel type";
           }
           leaf payload-type {
               type enumeration {
                   enum ipv4 {
                       value 0;
                       description "IPv4";
                   }
                   enum ipv6 {
                       value 1;
                       description "IPv6";
                   }
                   enum dual {
                       value 2;
                       description "IPv4 and IPv6";
                   }
               }
               description "Payload Type";
           }
           leaf gre-key {
               type uint32;
               description "GRE_KEY";
           }
           container gtp-tunnel-info {
               leaf local-tunnel-identifier {
                   type uint32;
                   description "Tunnel Endpoint IDentifier (TEID)";
               }
               leaf remote-tunnel-identifier {
                   type uint32;
                   description "Tunnel Endpoint IDentifier (TEID)";
               }
               leaf sequence-numbers-enabled {
                   type boolean;
                   description "Sequence No. Enabled";
               }
               description "GTP Tunnel Information";
           }
           leaf ebi {
               type fpcbase:ebi-type;
               description "EPS Bearier Identifier";
           }
           leaf lbi {
               type fpcbase:ebi-type;
               description "Linked Bearier Identifier";
           }
           description "Tunnel Information";
         }
         description "Tunnel Value";
       }

     //////////////////////////////
     // DESCRIPTOR DEFINITIONS

       // From 3GPP TS 24.008 version 13.5.0 Release 13
         typedef packet-filter-direction {
             type enumeration {
               enum preRel7Tft {
                 value 0;
                 description "Pre-Release 7 TFT";
               }
               enum uplink {
                 value 1;
                 description "uplink";
               }
               enum downlink {
                 value 2;
                 description "downlink";
               }
               enum bidirectional {
                 value 3;
                 description "bi-direcitonal";
               }
             }
             description "Packet Filter Direction";
         }
         typedef component-type-id {
             type uint8 {
               range "16 | 17 | 32 | 33 | 35 | 48 | 64 | 65 |"
               + " 80 | 81 | 96 | 112 | 128";
             }
             description "Specifies the Component Type";
         }
         grouping packet-filter {
           leaf direction {
               type fpcbase:packet-filter-direction;
               description "Filter Direction";
           }
           leaf identifier {
               type uint8 {
                 range "1..15";
               }
               description "Filter Identifier";
           }
           leaf evaluation-precedence {
               type uint8;
               description "Evaluation Precedence";
           }
           list contents {
             key component-type-identifier;
             description "Filter Contents";
             leaf component-type-identifier {
                 type fpcbase:component-type-id;
                 description "Component Type";
             }
             choice value {
               leaf ipv4-local {
                   type inet:ipv4-address;
                   description "IPv4 Local Address";
               }
               leaf ipv6-prefix-local {
                   type inet:ipv6-prefix;
                   description "IPv6 Local Prefix";
               }
               leaf ipv4-ipv6-remote {
                   type inet:ip-address;
                   description "Ipv4 Ipv6 remote address";
               }
               leaf ipv6-prefix-remote {
                   type inet:ipv6-prefix;
                   description "IPv6 Remote Prefix";
               }
               leaf next-header {
                   type uint8;
                   description "Next Header";
               }
               leaf local-port {
                   type inet:port-number;
                   description "Local Port";
               }
               case local-port-range {
                 leaf local-port-lo {
                   type inet:port-number;
                   description "Local Port Min Value";
                 }
                 leaf local-port-hi {
                   type inet:port-number;
                   description "Local Port Max Value";
                 }
               }
               leaf remote-port {
                   type inet:port-number;
                   description "Remote Port";
               }
               case remote-port-range {
                 leaf remote-port-lo {
                   type inet:port-number;
                   description "Remote Por Min Value";
                 }
                 leaf remote-port-hi {
                   type inet:port-number;
                   description "Remote Port Max Value";
                 }
               }
               leaf ipsec-index {
                   type traffic-selectors:ipsec-spi;
                   description "IPSec Index";
               }
               leaf traffic-class {
                   type inet:dscp;
                   description "Traffic Class";
               }
               case traffic-class-range {
                   leaf traffic-class-lo {
                     type inet:dscp;
                     description "Traffic Class Min Value";
                   }
                   leaf traffic-class-hi {
                     type inet:dscp;
                     description "Traffic Class Max Value";
                   }
               }
               leaf-list flow-label {
                   type inet:ipv6-flow-label;
                   description "Flow Label";
               }
               description "Component Value";
             }
           }
           description "Packet Filter";
         }

       grouping prefix-descriptor {
           leaf destination-ip {
             type inet:ip-prefix;
             description "Rule of destination IP";
           }
           leaf source-ip {
             type inet:ip-prefix;
             description "Rule of source IP";
           }
           description "Traffic descriptor based upon source/
             destination as IP prefixes";
       }

     grouping fpc-descriptor-value {
       choice descriptor-value {
         mandatory true;
         leaf all-traffic {
             type empty;
             description "admit any";
         }
         leaf no-traffic {
             type empty;
```

```
               description "deny any";
             }
             case prefix-descriptor {
               uses fpcbase:prefix-descriptor;
               description "IP Prefix descriptor";
             }
             case pmip-selector {
               uses traffic-selectors:traffic-selector;
               description "PMIP Selector";
             }
             container rfc5777-classifier-template {
                   uses rfc5777:classifier;
                   description "RFC 5777 Classifier";
             }
             container packet-filter {
                   uses fpcbase:packet-filter;
                   description "Packet Filter";
             }
             case tunnel-info {
               uses fpcbase:tunnel-value;
               description "Tunnel Descriptor (only
                   considers source info)";
             }
             description "Descriptor Value";
           }
           description "FPC Descriptor Values";
         }

           // Next Hop Structures
           typedef fpc-service-path-id {
               type uint32 {
                   range "0..33554431";
               }
               description "SERVICE_PATH_ID";
           }
           typedef fpc-mpls-label {
               type uint32 {
                 range "0..1048575";
               }
               description "MPLS label";
           }
           typedef segment-id {
               type string {
                   length "16";
               }
               description "SR Segement Identifier";
           }
           grouping fpc-nexthop {
```

```
           choice next-hop-value {
               leaf ip-address {
                   type inet:ip-address;
                   description "IP Value";
               }
               leaf mac-address {
                   type ytypes:mac-address;
                   description "MAC Address Value";
               }
               leaf service-path {
                   type fpcbase:fpc-service-path-id;
                   description "Service Path Value";
               }
               leaf mpls-path {
                   type fpcbase:fpc-mpls-label;
                   description "MPLS Value";
               }
               leaf nsh {
                   type string {
                       length "16";
                   }
                   description "Network Service Header";
               }
               leaf interface {
                   type uint16;
                   description "If (interface) Value";
               }
               leaf segment-identifier {
                   type fpcbase:segment-id;
                   description "Segment Id";
               }
               leaf-list mpls-label-stack {
                   type fpcbase:fpc-mpls-label;
                   description "MPLS Stack";
               }
               leaf-list mpls-sr-stack {
                   type fpcbase:fpc-mpls-label;
                   description "MPLS SR Stack";
               }
               leaf-list srv6-stack {
                   type fpcbase:segment-id;
                   description "Segment Id";
               }
               case tunnel-info {
                   uses fpcbase:tunnel-value;
                   description "Tunnel Descriptor (only
                   considers source info)";
               }
               description "Value";
           }
           description "Nexthop Value";
       }

       //////////////////////////////
       // PMIP Integration         //
         typedef pmip-commandset {
               type bits {
                   bit assign-ip {
                     position 0;
                     description "Assign IP";
                   }
                   bit assign-dpn {
                     position 1;
                     description "Assign DPN";
                   }
                   bit session {
                     position 2;
                     description "Session Level";
                   }
                   bit uplink {
                     position 3;
                     description "Uplink";
                   }
                   bit downlink {
                     position 4;
                     description "Downlink";
                   }
               }
               description "PMIP Instructions";
           }
       ///////////////////////////////
       // 3GPP Integration         //

         // Type Defs
         typedef fpc-qos-class-identifier {
             type uint8 {
                 range "1..9";
             }
             description "QoS Class Identifier (QCI)";
         }
         typedef ebi-type {
           type uint8 {
             range "0..15";
           }
           description "EUTRAN Bearere Identifier (EBI) Type";
         }
         typedef imsi-type {
             type uint64;
             description
                 "International Mobile Subscriber Identity (IMSI)
                   Value Type";
         }
         // Instructions
         typedef threegpp-instr {
           type bits {
             bit assign-ip {
               position 0;
               description "Assign IP Address/Prefix";
             }
             bit assign-fteid-ip {
               position 1;
               description "Assign FTEID-IP";
             }
             bit assign-fteid-teid {
               position 2;
               description "Assign FTEID-TEID";
             }
             bit session {
               position 3;
               description "Commands apply to the Session Level";
             }
             bit uplink {
               position 4;
               description "Commands apply to the Uplink";
             }
             bit downlink {
               position 5;
               description "Commands apply to the Downlink";
             }
             bit assign-dpn {
               position 6;
               description "Assign DPN";
             }
           }
           description "Instruction Set for 3GPP R11";
         }

       ///////////////////////////////
       // ACTION VALUE AUGMENTS
       grouping fpc-action-value {
           choice action-value {
               mandatory true;
               leaf drop {
                   type empty;
```

```
                        description "Drop Traffic";
                    }
                    container rewrite {
                        choice rewrite-value {
                            case prefix-descriptor {
                                uses fpcbase:prefix-descriptor;
                                description "IP Prefix descriptor";
                            }
                            case pmip-selector {
                                uses traffic-selectors:traffic-selector;
                                description "PMIP Selector";
                            }
                            container rfc5777-classifier-template {
                                uses rfc5777:classifier;
                                description "RFC 5777 Classifier";
                            }
                            description "Rewrite Choice";
                        }
                        description "Rewrite/NAT value";
                    }
                    container copy-forward-nexthop {
                            uses fpcbase:fpc-nexthop;
                            description "Copy Forward Value";
                    }
                    container nexthop {
                            uses fpcbase:fpc-nexthop;
                            description "NextHop Value";
                    }
                    case qos {
                        leaf trafficclass {
                            type inet:dscp;
                            description "Traffic Class";
                        }
                        uses pmipqos:qosattribute;
                        leaf qci {
                            type fpcbase:fpc-qos-class-identifier;
                            description "QCI";
                        }
                        leaf ue-agg-max-bitrate {
                            type uint32;
                            description "UE Aggregate Max Bitrate";
                        }
                        leaf apn-ambr {
                            type uint32;
                            description
                             "Access Point Name Aggregate Max Bit Rate";
                        }
                        description "QoS Attributes";
```

```
               }
               description "Action Value";
           }
           description "FPC Action Value";
       }

     // Instructions
     grouping instructions {
       container command-set {
         choice instr-type {
           leaf instr-3gpp-mob {
               type fpcbase:threegpp-instr;
               description "3GPP GTP Mobility Instructions";
           }
           leaf instr-pmip {
               type pmip-commandset;
               description "PMIP Instructions";
           }
           description "Instruction Value Choice";
         }
         description "Instructions";
       }
       description "Instructions Value";
     }
   }
   <CODE ENDS>
```

A.3. PMIP QoS Model

This module defines the base protocol elements specified in this document.

This module references [RFC6991].

```
   <CODE BEGINS> file "ietf-pmip-qos@2018-05-17.yang"
   module ietf-pmip-qos {
       yang-version 1.1;

       namespace
         "urn:ietf:params:xml:ns:yang:ietf-pmip-qos";

       prefix "qos-pmip";

       import ietf-inet-types {
         prefix inet;
         revision-date 2013-07-15;
       }
       import ietf-trafficselector-types { prefix traffic-selectors;
```

```
           revision-date 2018-05-17; }

       organization "IETF Distributed Mobility Management (DMM)
         Working Group";

       contact
          "WG Web:   <http://tools.ietf.org/wg/netmod/>
           WG List:  <mailto:netmod@ietf.org>

           WG Chair: Dapeng Liu
                     <mailto:maxpassion@gmail.com>

           WG Chair: Sri Gundavelli
                     <mailto:sgundave@cisco.com>

           Editor:   Satoru Matsushima
                     <mailto:satoru.matsushima@g.softbank.co.jp>

           Editor:   Lyle Bertz
                     <mailto:lylebe551144@gmail.com>";

       description
         "This module contains a collection of YANG definitions for
        quality of service paramaters used in Proxy Mobile IPv6.

        Copyright (c) 2016 IETF Trust and the persons identified as the
        document authors. All rights reserved.

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

     revision 2018-05-17 {
       description "Initial Revision.";
       reference "RFC 6088: Traffic Selectors for Flow Bindings";
     }

       // Type Definitions

       // QoS Option Field Type Definitions
     typedef sr-id {
       type uint8;
```

```
         description
          "An 8-bit unsigned integer used for identifying the QoS
           Service Request.";
       }

       typedef traffic-class {
         type inet:dscp;
         description
           "Traffic Class consists of a 6-bit DSCP field followed by a
            2-bit reserved field.";
        reference
            "RFC 3289: Management Information Base for the
               Differentiated Services Architecture
             RFC 2474: Definition of the Differentiated Services Field
                       (DS Field) in the IPv4 and IPv6 Headers
             RFC 2780: IANA Allocation Guidelines For Values In
                       the Internet Protocol and Related Headers";
       }

       typedef operational-code {
         type enumeration {
           enum RESPONSE {
             value 0;
             description "Response to a QoS request";
           }
           enum ALLOCATE {
             value 1;
             description "Request to allocate QoS resources";
           }
           enum DE-ALLOCATE {
             value 2;
             description "Request to de-Allocate QoS resources";
           }
           enum MODIFY {
             value 3;
             description "Request to modify QoS parameters for a
                 previously negotiated QoS Service Request";
           }
           enum QUERY {
             value 4;
             description "Query to list the previously negotiated QoS
                 Service Requests that are still active";
           }
           enum NEGOTIATE {
             value 5;
             description "Response to a QoS Service Request with a
               counter QoS proposal";
           }
         }
         description
          "The type of QoS request. Reserved values:   (6) to (255)
                   Currently not used.  Receiver MUST ignore the option
                   received with any value in this range.";
       }

     //Value definitions
     typedef Per-MN-Agg-Max-DL-Bit-Rate-Value {
         type uint32;
         description
             "The aggregate maximum downlink bit rate that is
             requested/allocated for all the mobile node's IP flows.
             The measurement units are bits per second.";
     }

      typedef Per-MN-Agg-Max-UL-Bit-Rate-Value {
         type uint32;
         description
           "The aggregate maximum uplink bit rate that is
               requested/allocated for the mobile node's IP flows.  The
               measurement units are bits per second.";
      }

      // Generic Structure for the uplink and downlink
      grouping Per-Session-Agg-Max-Bit-Rate-Value {
        leaf max-rate {
          type uint32;
          mandatory true;
          description
          "The aggregate maximum bit rate that is requested/allocated
        for all the IP flows associated with that mobility session.
        The measurement units are bits per second.";
        }
        leaf service-flag {
         type boolean;
         mandatory true;
         description
          "This flag is used for extending the scope of the
           target flows for Per-Session-Agg-Max-UL/DL-Bit-Rate
           from(UL)/to(DL) the mobile node's other mobility sessions
           sharing the same Service Identifier.";
         reference
           "RFC 5149 - Service Selection mobility option";
        }
        leaf exclude-flag {
          type boolean;
          mandatory true;
          description
           "This flag is used to request that the uplink/downlink
          flows for which the network is providing
               Guaranteed-Bit-Rate service be excluded from the
               target IP flows for which
               Per-Session-Agg-Max-UL/DL-Bit-Rate is measured.";
        }
       description "Per-Session-Agg-Max-Bit-Rate Value";
      }

      grouping Allocation-Retention-Priority-Value {
        leaf priority-level {
          type uint8 {
            range "0..15";
          }
          mandatory true;
          description
          "This is a 4-bit unsigned integer value. It is used to decide
           whether a mobility session establishment or modification
           request can be accepted; this is typically used for
           admission control of Guaranteed Bit Rate traffic in case of
           resource limitations.";
        }
        leaf preemption-capability {
          type enumeration {
           enum enabled {
             value 0;
             description "enabled";
           }
           enum disabled {
             value 1;
             description "disabled";
           }
           enum reserved1 {
             value 2;
             description "reserved1";
           }
           enum reserved2 {
             value 3;
             description "reserved2";
           }
          }
          mandatory true;
          description
          "This is a 2-bit unsigned integer value.  It defines whether a
           service data flow can get resources tha were already
           assigned to another service data flow with a lower priority
           level.";
        }
        leaf preemption-vulnerability {
          type enumeration {
           enum enabled {
             value 0;
             description "enabled";
           }
           enum disabled {
             value 1;
             description "disabled";
           }
           enum reserved1 {
             value 2;
             description "reserved1";
           }
           enum reserved2 {
             value 3;
             description "reserved2";
           }
          }
          mandatory true;
          description
          "This is a 2-bit unsigned integer value.  It defines whether a
            service data flow can lose the resources assigned to it in
            order to admit a service data flow with a higher priority
            level.";
        }
       description "Allocation-Retention-Priority Value";
      }

      typedef Aggregate-Max-DL-Bit-Rate-Value {
         type uint32;
         description
           "The aggregate maximum downlink bit rate that is
            requested/allocated for downlink IP flows.  The measurement
            units are bits per second.";
      }

       typedef Aggregate-Max-UL-Bit-Rate-Value {
         type uint32;
         description
           "The aggregate maximum downlink bit rate that is
            requested/allocated for downlink IP flows.  The measurement
            units are bits per second.";
       }

       typedef Guaranteed-DL-Bit-Rate-Value {
         type uint32;
         description
         "The guaranteed bandwidth in bits per second for downlink
           IP flows.  The measurement units are bits per second.";
       }

       typedef Guaranteed-UL-Bit-Rate-Value {
         type uint32;
         description
           "The guaranteed bandwidth in bits per second for uplink
            IP flows.  The measurement units are bits per second.";
       }

       grouping QoS-Vendor-Specific-Attribute-Value-Base {
         leaf vendorid {
           type uint32;
           mandatory true;
           description
            "The Vendor ID is the SMI (Structure of Management
             Information) Network Management Private Enterprise Code of
             the IANA-maintained ’Private Enterprise Numbers’
             registry.";
           reference
             "’PRIVATE ENTERPRISE NUMBERS’, SMI Network Management
               Private Enterprise Codes, April 2014,
                <http://www.iana.org/assignments/enterprise-numbers>";
         }
         leaf subtype {
           type uint8;
           mandatory true;
           description
             "An 8-bit field indicating the type of vendor-specific
              information carried in the option.  The namespace for this
              sub-type is managed by the vendor identified by the
              Vendor ID field.";
         }
         description
           "QoS Vendor-Specific Attribute.";
       }

       //Primary Structures (groupings)
       grouping qosattribute {
           leaf per-mn-agg-max-dl {
               type qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value;
               description "Per-MN-Agg-Max-DL-Bit-Rate Value";
           }
           leaf per-mn-agg-max-ul {
               type qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value;
               description "Per-MN-Agg-Max-UL-Bit-Rate Value";
           }
           container per-session-agg-max-dl {
               uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
               description "Per-Session-Agg-Max-Bit-Rate Value";
           }
           container per-session-agg-max-ul {
               uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
               description "Per-Session-Agg-Max-Bit-Rate Value";
           }
           uses qos-pmip:Allocation-Retention-Priority-Value;
           leaf agg-max-dl {
               type qos-pmip:Aggregate-Max-DL-Bit-Rate-Value;
               description "Aggregate-Max-DL-Bit-Rate Value";
           }
           leaf agg-max-ul {
               type qos-pmip:Aggregate-Max-UL-Bit-Rate-Value;
               description "Aggregate-Max-UL-Bit-Rate Value";
           }
           leaf gbr-dl {
               type qos-pmip:Guaranteed-DL-Bit-Rate-Value;
               description "Guaranteed-DL-Bit-Rate Value";
           }
           leaf gbr-ul {
               type qos-pmip:Guaranteed-UL-Bit-Rate-Value;
               description "Guaranteed-UL-Bit-Rate Value";
           }
           description "PMIP QoS Attributes. Note Vendor option
           is not a part of this grouping";
       }

       grouping qosoption {
           leaf srid {
               type sr-id;
               mandatory true;
               description "Service Request Identifier";
           }
           leaf trafficclass {
               type traffic-class;
               mandatory true;
               description "Traffic Class";
           }
           leaf operationcode {
               type operational-code;
               mandatory true;
               description "Operation Code";
           }
           uses qos-pmip:qosattribute;
           uses qos-pmip:QoS-Vendor-Specific-Attribute-Value-Base;
           container traffic-selector {
               uses traffic-selectors:traffic-selector;
               description "traffic selector";
           }
           description "PMIP QoS Option";
       }
   }
   <CODE ENDS>
```

A.4. Traffic Selectors YANG Model

This module defines traffic selector types commonly used in Proxy Mobile IP (PMIP).

This module references [RFC6991].

```
   <CODE BEGINS> file "ietf-trafficselector-types@2018-05-17.yang"
   module ietf-trafficselector-types {
    yang-version 1.1;

    namespace
    "urn:ietf:params:xml:ns:yang:ietf-trafficselector-types";

    prefix "traffic-selectors";

    import ietf-inet-types {
      prefix inet;
      revision-date 2013-07-15;
    }

    organization "IETF Distributed Mobility Management (DMM)
    Working Group";

    contact
    "WG Web: <http://tools.ietf.org/wg/netmod/>
    WG List: <mailto:netmod@ietf.org>

    WG Chair: Dapeng Liu
    <mailto:maxpassion@gmail.com>

    WG Chair: Sri Gundavelli
    <mailto:sgundave@cisco.com>

    Editor: Satoru Matsushima
    <mailto:satoru.matsushima@g.softbank.co.jp>

    Editor: Lyle Bertz
    <mailto:lylebe551144@gmail.com>";
    description
    "This module contains a collection of YANG definitions for
    traffic selectors for flow bindings.

    Copyright (c) 2016 IETF Trust and the persons identified as the
    document authors. All rights reserved.

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

     revision 2018-05-17 {
       description
        "Initial Revision.";
       reference
        "RFC 6088: Traffic Selectors for Flow Bindings";
     }

    // Identities
      identity traffic-selector-format {
        description
        "The base type for Traffic-Selector Formats";
      }

      identity ipv4-binary-selector-format {
        base traffic-selector-format;
        description
          "IPv4 Binary Traffic Selector Format";
      }

      identity ipv6-binary-selector-format {
        base traffic-selector-format;
        description
          "IPv6 Binary Traffic Selector Format";
      }

      // Type definitions and groupings
      typedef ipsec-spi {
        type uint32;
        description
         "The first 32-bit IPsec Security Parameter Index (SPI)
         value on data. This field is defined in [RFC4303].";
```

```
          reference
          "RFC 4303: IP Encapsulating Security
          Payload (ESP)";
      }

      grouping traffic-selector-base {
        description "A grouping of the commen leaves between the
          v4 and v6 Traffic Selectors";
        container ipsec-spi-range {
          presence "Enables setting ipsec spi range";
          description
          "Inclusive range representing IPSec Security Parameter
          Indices to be used. When only start-spi is present, it
          represents a single spi.";
      leaf start-spi {
          type ipsec-spi;
          mandatory true;
          description
            "The first 32-bit IPsec SPI value on data.";
          }
      leaf end-spi {
            type ipsec-spi;
            must ". >= ../start-spi" {
              error-message
                "The end-spi must be greater than or equal
                 to start-spi";
          }
         description
           "If more than one contiguous SPI value needs to be matched,
           then this field indicates the end value of a range.";
          }
       }
       container source-port-range {
         presence "Enables setting source port range";
         description
          "Inclusive range representing source ports to be used.
           When only start-port is present, it represents a single
        port. These value(s) are from the range of port numbers
           defined by IANA (http://www.iana.org).";
         leaf start-port {
            type inet:port-number;
            mandatory true;
            description
            "The first 16-bit source port number to be matched";
         }
         leaf end-port {
            type inet:port-number;
            must ". >= ../start-port" {
```

```
            error-message
              "The end-port must be greater than or equal to start-port";
           }
           description
            "The last 16-bit source port number to be matched";
          }
       }
       container destination-port-range {
         presence "Enables setting destination port range";
         description
          "Inclusive range representing destination ports to be used.
          When only start-port is present, it represents a single
          port.";
           leaf start-port {
             type inet:port-number;
             mandatory true;
             description
             "The first 16-bit destination port number to be matched";
          }
          leaf end-port {
            type inet:port-number;
            must ". >= ../start-port" {
            error-message
              "The end-port must be greater than or equal to
             start-port";
           }
           description
        "The last 16-bit destination port number to be matched";
        }
      }
    }

    grouping ipv4-binary-traffic-selector {
      container source-address-range-v4 {
         presence "Enables setting source IPv4 address range";
         description
          "Inclusive range representing IPv4 addresses to be used. When
          only start-address is present, it represents a single
          address.";
         leaf start-address {
           type inet:ipv4-address;
           mandatory true;
          description
           "The first source address to be matched";
         }
         leaf end-address {
           type inet:ipv4-address;
           description
```

```
             "The last source address to be matched";
           }
       }
       container destination-address-range-v4 {
          presence "Enables setting destination IPv4 address range";
          description
            "Inclusive range representing IPv4 addresses to be used.
            When only start-address is present, it represents a
            single address.";
          leaf start-address {
            type inet:ipv4-address;
            mandatory true;
            description
             "The first destination address to be matched";
          }
          leaf end-address {
            type inet:ipv4-address;
            description
             "The last destination address to be matched";
          }
       }
       container ds-range {
          presence "Enables setting dscp range";
          description
           "Inclusive range representing DiffServ Codepoints to be used.
           When only start-ds is present, it represents a single
           Codepoint.";
          leaf start-ds {
            type inet:dscp;
            mandatory true;
            description
             "The first differential service value to be matched";
        }
        leaf end-ds {
          type inet:dscp;
          must ". >= ../start-ds" {
            error-message
              "The end-ds must be greater than or equal to start-ds";
          }
          description
            "The last differential service value to be matched";
       }
      }
      container protocol-range {
        presence "Enables setting protocol range";
        description
          "Inclusive range representing IP protocol(s) to be used. When
           only start-protocol is present, it represents a single
          protocol.";
       leaf start-protocol {
         type uint8;
         mandatory true;
         description
           "The first 8-bit protocol value to be matched.";
        }
        leaf end-protocol {
          type uint8;
          must ". >= ../start-protocol" {
            error-message
              "The end-protocol must be greater than or equal to
             start-protocol";
          }
        description
          "The last 8-bit protocol value to be matched.";
        }
      }
      description "ipv4 binary traffic selector";
    }
     grouping ipv6-binary-traffic-selector {
      container source-address-range-v6 {
        presence "Enables setting source IPv6 address range";
         description
          "Inclusive range representing IPv6 addresses to be used.
          When only start-address is present, it represents a
          single address.";
         leaf start-address {
           type inet:ipv6-address;
           mandatory true;
           description
           "The first source address, from the
           range of 128-bit IPv6 addresses to be matched";
         }
         leaf end-address {
           type inet:ipv6-address;
           description
               "The last source address, from the
               range of 128-bit IPv6 addresses to be matched";
         }
      }
      container destination-address-range-v6 {
        presence "Enables setting destination IPv6 address range";
        description
          "Inclusive range representing IPv6 addresses to be used.
           When only start-address is present, it represents a
           single address.";
        leaf start-address {
          type inet:ipv6-address;
          mandatory true;
          description
              "The first destination address, from the
              range of 128-bit IPv6 addresses to be matched";
        }
        leaf end-address {
          type inet:ipv6-address;
          description
              "The last destination address, from the
              range of 128-bit IPv6 addresses to be matched";
        }
      }
      container flow-label-range {
        presence "Enables setting Flow Label range";
        description
          "Inclusive range representing IPv4 addresses to be used. When
           only start-flow-label is present, it represents a single
           flow label.";
        leaf start-flow-label {
          type inet:ipv6-flow-label;
          description
            "The first flow label value to be matched";
        }
        leaf end-flow-label {
          type inet:ipv6-flow-label;
          must ". >= ../start-flow-label" {
            error-message
              "The end-flow-lable must be greater than or equal to
               start-flow-label";
          }
          description
             "The first flow label value to be matched";
        }
       }
      container traffic-class-range {
        presence "Enables setting the traffic class range";
        description
         "Inclusive range representing IPv4 addresses to be used. When
          only start-traffic-class is present, it represents a single
          traffic class.";
        leaf start-traffic-class {
          type inet:dscp;
          description
           "The first traffic class value to be matched";
          reference
           "RFC 3260: New Terminology and Clarifications for Diffserv
            RFC 3168: The Addition of Explicit Congestion Notification
            (ECN) to IP";
        }
        leaf end-traffic-class {
          type inet:dscp;
          must ". >= ../start-traffic-class" {
            error-message
              "The end-traffic-class must be greater than or equal to
               start-traffic-class";
          }
          description
            "The last traffic class value to be matched";
        }
      }
      container next-header-range {
        presence "Enables setting Next Header range";
        description
         "Inclusive range representing Next Headers to be used. When
          only start-next-header is present, it represents a
          single Next Header.";
        leaf start-next-header {
          type uint8;
          description
           "The first 8-bit next header value to be matched.";
        }
        leaf end-next-header {
          type uint8;
          must ". >= ../start-next-header" {
            error-message
              "The end-next-header must be greater than or equal to
              start-next-header";
          }
          description
            "The last 8-bit next header value to be matched.";
        }
      }
      description "ipv6 binary traffic selector";
    }

      grouping traffic-selector {
        leaf ts-format {
           type identityref {
             base traffic-selector-format;
           }
           description "Traffic Selector Format";
        }
        uses traffic-selectors:traffic-selector-base;
        uses traffic-selectors:ipv4-binary-traffic-selector;
        uses traffic-selectors:ipv6-binary-traffic-selector;
```

```
       description
        "The traffic selector includes the parameters used to match
          packets for a specific flow binding.";
       reference
        "RFC 6089: Flow Bindings in Mobile IPv6 and Network
          Mobility (NEMO) Basic Support";
     }
   }
   <CODE ENDS>
```

A.5. RFC 5777 Classifier YANG Model

This module defines the RFC 5777 Classifer.

This module references [RFC5777].

```
   <CODE BEGINS> file "ietf-diam-trafficclassifier@2018-05-17.yang"
   module ietf-diam-trafficclassifier {
    yang-version 1.1;

    namespace
    "urn:ietf:params:xml:ns:yang:ietf-diam-trafficclassifier";

    prefix "diamclassifier";

    import ietf-inet-types {
      prefix inet;
      revision-date 2013-07-15;
    }
    import ietf-yang-types { prefix yang-types; }

    organization "IETF Distributed Mobility Management (DMM)
    Working Group";

    contact
    "WG Web: <http://tools.ietf.org/wg/netmod/>
    WG List: <mailto:netmod@ietf.org>

    WG Chair: Dapeng Liu
    <mailto:maxpassion@gmail.com>

    WG Chair: Sri Gundavelli
    <mailto:sgundave@cisco.com>

    Editor: Satoru Matsushima
    <mailto:satoru.matsushima@g.softbank.co.jp>

    Editor: Lyle Bertz
    <mailto:lylebe551144@gmail.com>";

    description
    "This module contains a collection of YANG definitions for
    traffic classification and QoS Attributes for Diameter.

    Copyright (c) 2018 IETF Trust and the persons identified as the
    document authors. All rights reserved.

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

     revision 2018-05-17 {
       description
           "Initial";
       reference
       "RFC 5777: Traffic Classification and Quality of Service (QoS)
           Attributes for Diameter";
     }

       typedef eui64-address-type {
           type string {
               length "6";
           }
           description
               "specifies a single layer 2 address in EUI-64 format.
                The value is an 8-octet encoding of the address as
                it would appear in the frame header.";
       }
       typedef direction-type {
           type enumeration {
               enum IN {
                   value 0;
                   description
                       "Applies to flows from the managed terminal.";
               }
               enum OUT {
                   value 1;
                   description
                    "Applies to flows to the managed terminal.";
               }
               enum BOTH {
                   value 2;
                   description
                       "Applies to flows both to and from the managed
                           terminal.";
               }
           }
           description
               "Specifies in which direction to apply the classifier.";
       }
       typedef negated-flag-type {
           type enumeration {
               enum False { value 0;
                   description "false"; }
               enum True { value 1;
                   description "True"; }
           }
           description
               "When set to True, the meaning of the match is
                inverted and the classifier will match addresses
                other than those specified by the From-Spec or
                To-Spec AVP.

                Note that the negation does not impact the port
                   comparisons.";
       }
       grouping index {
           leaf index {
               type uint16;
               mandatory true;
               description "Identifier used for referencing";
           }
           description "Index Value";
       }
       grouping to-from-spec-value {
           leaf-list ip-address {
               type inet:ip-address;
               description "IP address";
           }
           list ip-address-range {
               key index;
               uses diamclassifier:index;
               leaf ip-address-start {
                   type inet:ip-address;
                   description "IP Address Start";
               }
               leaf ip-address-end {
                   type inet:ip-address;
                   description "IP Address End";
               }
               description "IP Address Range";
           }
           leaf-list ip-address-mask {
               type inet:ip-prefix;
               description "IP Address Mask";
           }
           leaf-list mac-address {
               type yang-types:mac-address;
               description "MAC address";
           }
           list mac-address-mask {
               key mac-address;
               leaf mac-address {
                   type yang-types:mac-address;
                   mandatory true;
                   description "MAC address";
               }
               leaf macaddress-mask-pattern {
                   type yang-types:mac-address;
                   mandatory true;
                   description
                    "The value specifies the bit positions of a
                     MAC address that are taken for matching.";
               }
               description "MAC Address Mask";
           }
           leaf-list eui64-address {
               type diamclassifier:eui64-address-type;
               description "EUI64 Address";
           }
           list eui64-address-mask {
               key eui64-address;
               leaf eui64-address {
                   type diamclassifier:eui64-address-type;
                   mandatory true;
                   description "eui64 address";
               }
               leaf eui64-address-mask-pattern {
                   type diamclassifier:eui64-address-type;
                   mandatory true;
                   description
                    "The value is 8 octets specifying the bit
                     positions of a EUI64 address that are taken
                    for matching.";
               }
               description "EUI64 Address Mask";
           }
           leaf-list port {
               type inet:port-number;
               description "Port Number";
           }
           list port-range {
               key index;
               uses diamclassifier:index;
               leaf ip-address-start {
                   type inet:port-number;
                   description "Port Start";
               }
               leaf ip-address-end {
                   type inet:port-number;
                   description "Port End";
               }
               description "Port Range";
           }
           leaf negated {
               type diamclassifier:negated-flag-type;
               description "Negated";
           }
           leaf use-assigned-address {
               type boolean;
               description "Use Assigned Address";
           }
           description
               "Basic traffic description value";
       }

       grouping option-type-group {
           leaf option-type {
               type uint8;
               mandatory true;
               description "Option Type";
           }
           leaf-list ip-option-value {
               type string;
               description "Option Value";
           }
           leaf negated {
               type diamclassifier:negated-flag-type;
               description "Negated";
           }
           description "Common X Option Pattern";
       }
       typedef vlan-id {
           type uint32 {
               range "0..4095";
           }
           description "VLAN ID";
       }

     grouping classifier {
       leaf protocol {
           type uint8;
           description "Protocol";
       }
       leaf direction {
           type diamclassifier:direction-type;
           description "Direction";
       }
       list from-spec {
           key index;
           uses diamclassifier:index;
           uses diamclassifier:to-from-spec-value;
           description "from specification";
       }
       list to-spec {
           key index;
           uses diamclassifier:index;
           uses diamclassifier:to-from-spec-value;
           description "to specification";
       }
       leaf-list disffserv-code-point {
           type inet:dscp;
           description "DSCP";
       }
       leaf fragmentation-flag {
       type enumeration {
               enum DF {
                   value 0;
                   description "Don’t Fragment";
               }
               enum MF {
                   value 1;
                   description "More Fragments";
               }
           }
           description "Fragmenttation Flag";
       }
       list ip-option {
           key option-type;
           uses diamclassifier:option-type-group;
           description "IP Option Value";
       }
```

```
       list tcp-option {
           key option-type;
           uses diamclassifier:option-type-group;
           description "TCP Option Value";
       }
       list tcp-flag {
           key tcp-flag-type;
           leaf tcp-flag-type {
               type uint32;
               mandatory true;
               description "TCP Flag Type";
           }
           leaf negated {
               type diamclassifier:negated-flag-type;
               description "Negated";
           }
           description "TCP Flags";
       }
       list icmp-option {
           key option-type;
           uses diamclassifier:option-type-group;
           description "ICMP Option Value";
       }
       list eth-option {
           key index;
           uses diamclassifier:index;
           container eth-proto-type {
               leaf-list eth-ether-type {
                   type string {
                       length "2";
                   }
                   description "value of ethertype field";
               }
               leaf-list eth-sap {
                   type string {
                       length "2";
                   }
                   description "802.2 SAP";
               }
               description "Ether Proto Type";
           }
           list vlan-id-range {
               key index;
               uses diamclassifier:index;
               leaf-list s-vlan-id-start {
                   type diamclassifier:vlan-id;
                   description "S-VID  VLAN ID Start";
               }
               leaf-list s-vlan-id-end {
                   type diamclassifier:vlan-id;
                   description "S-VID VLAN ID End";
               }
               leaf-list c-vlan-id-start {
                   type diamclassifier:vlan-id;
                   description "C-VID  VLAN ID Start";
               }
               leaf-list c-vlan-id-end {
                   type diamclassifier:vlan-id;
                   description "C-VID  VLAN ID End";
               }
               description "VLAN ID Range";
           }
           list user-priority-range {
               key index;
               uses diamclassifier:index;
               leaf-list low-user-priority {
                   type uint32 {
                       range "0..7";
                   }
                   description "Low User Priority";
               }
               leaf-list high-user-priority {
                   type uint32 {
                       range "0..7";
                   }
                   description "High User Priority";
               }
               description "User priority range";
           }
           description "Ether Option";
       }
       description "RFC 5777 Classifier";
     }
   }
   <CODE ENDS>
```

Appendix B. FPC YANG Tree Structure

This section only shows the structure for FPC YANG model. NOTE, it does NOT show the settings, Action values or Descriptor Value.

```
   descriptor_value:
   +--rw (descriptor-value)
     +--:(all-traffic)
     |  +--rw all-traffic?                    empty
     +--:(no-traffic)
     |  +--rw no-traffic?                        empty
     +--:(prefix-descriptor)
     |  +--rw destination-ip?                    inet:ip-prefix
     |  +--rw source-ip?                         inet:ip-prefix
     +--:(pmip-selector)
     |  +--rw ts-format?                         identityref
     |  +--rw ipsec-spi-range!
     |  |  +--rw start-spi    ipsec-spi
     |  |  +--rw end-spi?     ipsec-spi
     |  +--rw source-port-range!
     |  |  +--rw start-port    inet:port-number
     |  |  +--rw end-port?     inet:port-number
     |  +--rw destination-port-range!
     |  |  +--rw start-port    inet:port-number
     |  |  +--rw end-port?     inet:port-number
     |  +--rw source-address-range-v4!
     |  |  +--rw start-address    inet:ipv4-address
     |  |  +--rw end-address?     inet:ipv4-address
     |  +--rw destination-address-range-v4!
     |  |  +--rw start-address    inet:ipv4-address
     |  |  +--rw end-address?     inet:ipv4-address
     |  +--rw ds-range!
     |  |  +--rw start-ds    inet:dscp
     |  |  +--rw end-ds?     inet:dscp
     |  +--rw protocol-range!
     |  |  +--rw start-protocol    uint8
     |  |  +--rw end-protocol?     uint8
     |  +--rw source-address-range-v6!
     |  |  +--rw start-address    inet:ipv6-address
     |  |  +--rw end-address?     inet:ipv6-address
     |  +--rw destination-address-range-v6!
     |  |  +--rw start-address    inet:ipv6-address
     |  |  +--rw end-address?     inet:ipv6-address
     |  +--rw flow-label-range!
     |  |  +--rw start-flow-label?   inet:ipv6-flow-label
     |  |  +--rw end-flow-label?     inet:ipv6-flow-label
     |  +--rw traffic-class-range!
     |  |  +--rw start-traffic-class?   inet:dscp
     |  |  +--rw end-traffic-class?     inet:dscp
     |  +--rw next-header-range!
     |     +--rw start-next-header?   uint8
     |     +--rw end-next-header?     uint8
     +--:(rfc5777-classifier-template)
     |  +--rw rfc5777-classifier-template
     |     +--rw protocol?              uint8
     |     +--rw direction?             diamclassifier:direction-type
     |     +--rw from-spec* [index]
     |     |  +--rw index                  uint16
     |     |  +--rw ip-address*              inet:ip-address
     |     |  +--rw ip-address-range* [index]
     |     |  |  +--rw index                inet:uint16
     |     |  |  +--rw ip-address-start?   inet:ip-address
     |     |  |  +--rw ip-address-end?     inet:ip-address
     |     |  +--rw ip-address-mask*        inet:ip-prefix
     |     |  +--rw mac-address*            yang-types:mac-address
     |     |  +--rw mac-address-mask* [mac-address]
     |     |  |  +--rw mac-address                yang-types:mac-address
     |     |  |  +--rw macaddress-mask-pattern  yang-types:mac-address
     |     |  +--rw eui64-address*
                          diamclassifier:eui64-address-type
     |     |  +--rw eui64-address-mask* [eui64-address]
     |     |  |  +--rw eui64-address
                          diamclassifier:eui64-address-type
     |     |  |  +--rw eui64-address-mask-pattern
                          diamclassifier:eui64-address-type
     |     |  +--rw port*                   inet:port-number
     |     |  +--rw port-range* [index]
     |     |  |  +--rw index                uint16
     |     |  |  +--rw ip-address-start?   inet:port-number
     |     |  |  +--rw ip-address-end?     inet:port-number
     |     |  +--rw negated?
                          diamclassifier:negated-flag-type
     |     |  +--rw use-assigned-address?   boolean
     |     +--rw to-spec* [index]
     |     |  +--rw index                   uint16
     |     |  +--rw ip-address*             inet:ip-address
     |     |  +--rw ip-address-range* [index]
     |     |  |  +--rw index                uint16
     |     |  |  +--rw ip-address-start?   inet:ip-address
     |     |  |  +--rw ip-address-end?     inet:ip-address
     |     |  +--rw ip-address-mask*        inet:ip-prefix
     |     |  +--rw mac-address*            yang-types:mac-address
     |     |  +--rw mac-address-mask* [mac-address]
     |     |  |  +--rw mac-address                  yang-types:mac-address
     |     |  |  +--rw macaddress-mask-pattern    yang-types:mac-address
     |     |  +--rw eui64-address*
                          diamclassifier:eui64-address-type
     |     |  +--rw eui64-address-mask* [eui64-address]
     |     |  |  +--rw eui64-address
                          diamclassifier:eui64-address-type
     |     |  |  +--rw eui64-address-mask-pattern
                          diamclassifier:eui64-address-type
     |     |  +--rw port*                   inet:port-number
     |     |  +--rw port-range* [index]
     |     |  |  +--rw index                uint16
     |     |  |  +--rw ip-address-start?   inet:port-number
     |     |  |  +--rw ip-address-end?     inet:port-number
     |     |  +--rw negated?
                          diamclassifier:negated-flag-type
     |     |  +--rw use-assigned-address?   boolean
     |     +--rw disffserv-code-point*   inet:dscp
     |     +--rw fragmentation-flag?     enumeration
     |     +--rw ip-option* [option-type]
     |     |  +--rw option-type        uint8
     |     |  +--rw ip-option-value*   string
     |     |  +--rw negated?           diamclassifier:negated-flag-type
     |     +--rw tcp-option* [option-type]
     |     |  +--rw option-type        uint8
     |     |  +--rw ip-option-value*   string
     |     |  +--rw negated?           diamclassifier:negated-flag-type
     |     +--rw tcp-flag* [tcp-flag-type]
     |     |  +--rw tcp-flag-type    uint32
     |     |  +--rw negated?         diamclassifier:negated-flag-type
     |     +--rw icmp-option* [option-type]
     |     |  +--rw option-type        uint8
     |     |  +--rw ip-option-value*   string
     |     |  +--rw negated?           diamclassifier:negated-flag-type
     |     +--rw eth-option* [index]
     |        +--rw index                  uint16
     |        +--rw eth-proto-type
     |        |  +--rw eth-ether-type*   string
     |        |  +--rw eth-sap*          string
     |        +--rw vlan-id-range* [index]
     |        |  +--rw index              uint16
     |        |  +--rw s-vlan-id-start*   diamclassifier:vlan-id
     |        |  +--rw s-vlan-id-end*     diamclassifier:vlan-id
     |        |  +--rw c-vlan-id-start*   diamclassifier:vlan-id
     |        |  +--rw c-vlan-id-end*     diamclassifier:vlan-id
     |        +--rw user-priority-range* [index]
     |           +--rw index                 uint16
     |           +--rw low-user-priority*    uint32
     |           +--rw high-user-priority*   uint32
     +--:(packet-filter)
     |  +--rw packet-filter
     |     +--rw direction?             fpcbase:packet-filter-direction
     |     +--rw identifier?              uint8
     |     +--rw evaluation-precedence?   uint8
     |     +--rw contents* [component-type-identifier]
     |        +--rw component-type-identifier fpcbase:component-type-id
     |        +--rw (value)?
     |           +--:(ipv4-local)
     |           |  +--rw ipv4-local?                 inet:ipv4-address
     |           +--:(ipv6-prefix-local)
     |           |  +--rw ipv6-prefix-local?          inet:ipv6-prefix
     |           +--:(ipv4-ipv6-remote)
     |           |  +--rw ipv4-ipv6-remote?            inet:ip-address
     |           +--:(ipv6-prefix-remote)
     |           |  +--rw ipv6-prefix-remote?          inet:ipv6-prefix
     |           +--:(next-header)
     |           |  +--rw next-header?                 uint8
     |           +--:(local-port)
     |           |  +--rw local-port?                  inet:port-number
     |           +--:(local-port-range)
     |           |  +--rw local-port-lo?               inet:port-number
     |           |  +--rw local-port-hi?               inet:port-number
     |           +--:(remote-port)
     |           |  +--rw remote-port?                 inet:port-number
     |           +--:(remote-port-range)
     |           |  +--rw remote-port-lo?              inet:port-number
     |           |  +--rw remote-port-hi?              inet:port-number
     |           +--:(ipsec-index)
     |           |  +--rw ipsec-index?     traffic-selectors:ipsec-spi
     |           +--:(traffic-class)
     |           |  +--rw traffic-class?               inet:dscp
     |           +--:(traffic-class-range)
     |           |  +--rw traffic-class-lo?            inet:dscp
     |           |  +--rw traffic-class-hi?            inet:dscp
     |           +--:(flow-label)
     |              +--rw flow-label*    inet:ipv6-flow-label
     +--:(tunnel-info)
        +--rw tunnel-info
           +--rw tunnel-local-address?    inet:ip-address
           +--rw tunnel-remote-address?   inet:ip-address
           +--rw mtu-size?                uint32
           +--rw tunnel?                  identityref
           +--rw payload-type?            enumeration
           +--rw gre-key?                 uint32
           +--rw gtp-tunnel-info
           |  +--rw local-tunnel-identifier?    uint32
           |  +--rw remote-tunnel-identifier?   uint32
           |  +--rw sequence-numbers-enabled?   boolean
           +--rw ebi?                     fpcbase:ebi-type
           +--rw lbi?                     fpcbase:ebi-type

   action_value:
   +--:(action-value)
   |  +--rw (action-value)
   |     +--:(drop)
   |     |  +--rw drop?                             empty
   |     +--:(rewrite)
   |     |  +--rw rewrite
   |     |     +--rw (rewrite-value)?
   |     |        +--:(prefix-descriptor)
   |     |        |  +--rw destination-ip?              inet:ip-prefix
   |     |        |  +--rw source-ip?                   inet:ip-prefix
   |     |        +--:(pmip-selector)
   |     |        |  +--rw ts-format?                     identityref
   |     |        |  +--rw ipsec-spi-range!
   |     |        |  |  +--rw start-spi    ipsec-spi
   |     |        |  |  +--rw end-spi?     ipsec-spi
   |     |        |  +--rw source-port-range!
   |     |        |  |  +--rw start-port    inet:port-number
   |     |        |  |  +--rw end-port?     inet:port-number
   |     |        |  +--rw destination-port-range!
   |     |        |  |  +--rw start-port    inet:port-number
   |     |        |  |  +--rw end-port?     inet:port-number
   |     |        |  +--rw source-address-range-v4!
   |     |        |  |  +--rw start-address    inet:ipv4-address
   |     |        |  |  +--rw end-address?     inet:ipv4-address
   |     |        |  +--rw destination-address-range-v4!
   |     |        |  |  +--rw start-address    inet:ipv4-address
   |     |        |  |  +--rw end-address?     inet:ipv4-address
   |     |        |  +--rw ds-range!
   |     |        |  |  +--rw start-ds    inet:dscp
   |     |        |  |  +--rw end-ds?     inet:dscp
   |     |        |  +--rw protocol-range!
   |     |        |  |  +--rw start-protocol    uint8
   |     |        |  |  +--rw end-protocol?     uint8
   |     |        |  +--rw source-address-range-v6!
   |     |        |  |  +--rw start-address    inet:ipv6-address
   |     |        |  |  +--rw end-address?     inet:ipv6-address
   |     |        |  +--rw destination-address-range-v6!
   |     |        |  |  +--rw start-address    inet:ipv6-address
   |     |        |  |  +--rw end-address?     inet:ipv6-address
   |     |        |  +--rw flow-label-range!
   |     |        |  |  +--rw start-flow-label?   inet:ipv6-flow-label
   |     |        |  |  +--rw end-flow-label?     inet:ipv6-flow-label
   |     |        |  +--rw traffic-class-range!
   |     |        |  |  +--rw start-traffic-class?   inet:dscp
   |     |        |  |  +--rw end-traffic-class?     inet:dscp
   |     |        |  +--rw next-header-range!
   |     |        |     +--rw start-next-header?   uint8
   |     |        |     +--rw end-next-header?     uint8
   |     |        +--:(rfc5777-classifier-template)
   |     |           +--rw rfc5777-classifier-template
   |     |              +--rw protocol?               uint8
   |     |              +--rw direction?
                         diamclassifier:direction-type
   |     |              +--rw from-spec* [index]
   |     |              |  +--rw index                   uint16
   |     |              |  +--rw ip-address*             inet:ip-address
   |     |              |  +--rw ip-address-range* [index]
   |     |              |  |  +--rw index               uint16
   |     |              |  |  +--rw ip-address-start?   inet:ip-address
   |     |              |  |  +--rw ip-address-end?     inet:ip-address
   |     |              |  +--rw ip-address-mask*        inet:ip-prefix
   |     |              |  +--rw mac-address*    yang-types:mac-address
   |     |              |  +--rw mac-address-mask* [mac-address]
   |     |              |  |  +--rw mac-address
   |     |                          yang-types:mac-address
   |     |              |  |  +--rw macaddress-mask-pattern
   |     |                          yang-types:mac-address
   |     |              |  +--rw eui64-address*
   |     |                          diamclassifier:eui64-address-type
   |     |              |  +--rw eui64-address-mask* [eui64-address]
   |     |              |  |  +--rw eui64-address
   |     |                          diamclassifier:eui64-address-type
   |     |              |  |  +--rw eui64-address-mask-pattern
   |     |                          diamclassifier:eui64-address-type
   |     |              |  +--rw port*                 inet:port-number
   |     |              |  +--rw port-range* [index]
   |     |              |  |  +--rw index               uint16
   |     |              |  |  +--rw ip-address-start?   inet:port-number
   |     |              |  |  +--rw ip-address-end?     inet:port-number
   |     |              |  +--rw negated?
   |     |                          diamclassifier:negated-flag-type
   |     |              |  +--rw use-assigned-address?   boolean
   |     |              +--rw to-spec* [index]
   |     |              |  +--rw index                   uint16
   |     |              |  +--rw ip-address*             inet:ip-address
   |     |              |  +--rw ip-address-range* [index]
   |     |              |  |  +--rw index               uint16
   |     |              |  |  +--rw ip-address-start?   inet:ip-address
   |     |              |  |  +--rw ip-address-end?     inet:ip-address
   |     |              |  +--rw ip-address-mask*        inet:ip-prefix
   |     |              |  +--rw mac-address*
   |     |                          yang-types:mac-address
   |     |              |  +--rw mac-address-mask* [mac-address]
   |     |              |  |  +--rw mac-address
   |     |                          yang-types:mac-address
   |     |              |  |  +--rw macaddress-mask-pattern
   |     |                          yang-types:mac-address
   |     |              |  +--rw eui64-address*
   |     |                          diamclassifier:eui64-address-type
   |     |              |  +--rw eui64-address-mask* [eui64-address]
   |     |              |  |  +--rw eui64-address
   |     |                          diamclassifier:eui64-address-type
   |     |              |  |  +--rw eui64-address-mask-pattern
```

```
                                   diamclassifier:eui64-address-type
   |     |              |  +--rw port*                 inet:port-number
   |     |              |  +--rw port-range* [index]
   |     |              |  |  +--rw index               uint16
   |     |              |  |  +--rw ip-address-start?   inet:port-number
   |     |              |  |  +--rw ip-address-end?     inet:port-number
   |     |              |  +--rw negated?
                                   diamclassifier:negated-flag-type
   |     |              |  +--rw use-assigned-address?   boolean
   |     |              +--rw disffserv-code-point*   inet:dscp
   |     |              +--rw fragmentation-flag?     enumeration
   |     |              +--rw ip-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                   diamclassifier:negated-flag-type
   |     |              +--rw tcp-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                   diamclassifier:negated-flag-type
   |     |              +--rw tcp-flag* [tcp-flag-type]
   |     |              |  +--rw tcp-flag-type    uint32
   |     |              |  +--rw negated?
                                   diamclassifier:negated-flag-type
   |     |              +--rw icmp-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                   diamclassifier:negated-flag-type
   |     |              +--rw eth-option* [index]
   |     |                 +--rw index                  uint16
   |     |                 +--rw eth-proto-type
   |     |                 |  +--rw eth-ether-type*   string
   |     |                 |  +--rw eth-sap*          string
   |     |                 +--rw vlan-id-range* [index]
   |     |                 |  +--rw index              uint16
   |     |                 |  +--rw s-vlan-id-start*
                                   diamclassifier:vlan-id
   |     |                 |  +--rw s-vlan-id-end*
                                   diamclassifier:vlan-id
   |     |                 |  +--rw c-vlan-id-start*
                                   diamclassifier:vlan-id
   |     |                 |  +--rw c-vlan-id-end*
                                   diamclassifier:vlan-id
   |     |                 +--rw user-priority-range* [index]
   |     |                    +--rw index                 uint16
   |     |                    +--rw low-user-priority*    uint32
   |     |                    +--rw high-user-priority*   uint32
   |     +--:(copy-forward-nexthop)
   |     |  +--rw copy-forward-nexthop
   |     |     +--rw (next-hop-value)?
   |     |        +--:(ip-address)
   |     |        |  +--rw ip-address?           inet:ip-address
   |     |        +--:(mac-address)
   |     |        |  +--rw mac-address?          ytypes:mac-address
   |     |        +--:(service-path)
   |     |        |  +--rw service-path?   fpcbase:fpc-service-path-id
   |     |        +--:(mpls-path)
   |     |        |  +--rw mpls-path?            fpcbase:fpc-mpls-label
   |     |        +--:(nsh)
   |     |        |  +--rw nsh?                  string
   |     |        +--:(interface)
   |     |        |  +--rw interface?            uint16
   |     |        +--:(segment-identifier)
   |     |        |  +--rw segment-identifier?   fpcbase:segment-id
   |     |        +--:(mpls-label-stack)
   |     |        |  +--rw mpls-label-stack*     fpcbase:fpc-mpls-label
   |     |        +--:(mpls-sr-stack)
   |     |        |  +--rw mpls-sr-stack*        fpcbase:fpc-mpls-label
   |     |        +--:(srv6-stack)
   |     |        |  +--rw srv6-stack*           fpcbase:segment-id
   |     |        +--:(tunnel-info)
   |     |           +--rw tunnel-info
   |     |              +--rw tunnel-local-address?    inet:ip-address
   |     |              +--rw tunnel-remote-address?   inet:ip-address
   |     |              +--rw mtu-size?                uint32
   |     |              +--rw tunnel?                  identityref
   |     |              +--rw payload-type?            enumeration
   |     |              +--rw gre-key?                 uint32
   |     |              +--rw gtp-tunnel-info
   |     |              |  +--rw local-tunnel-identifier?    uint32
   |     |              |  +--rw remote-tunnel-identifier?   uint32
   |     |              |  +--rw sequence-numbers-enabled?   boolean
   |     |              +--rw ebi?                     fpcbase:ebi-type
   |     |              +--rw lbi?                     fpcbase:ebi-type
   |     +--:(nexthop)
   |     |  +--rw nexthop
   |     |     +--rw (next-hop-value)?
   |     |        +--:(ip-address)
   |     |        |  +--rw ip-address?           inet:ip-address
   |     |        +--:(mac-address)
   |     |        |  +--rw mac-address?          ytypes:mac-address
   |     |        +--:(service-path)
   |     |        |  +--rw service-path?   fpcbase:fpc-service-path-id
   |     |        +--:(mpls-path)
   |     |        |  +--rw mpls-path?            fpcbase:fpc-mpls-label
   |     |        +--:(nsh)
   |     |        |  +--rw nsh?                  string
   |     |        +--:(interface)
   |     |        |  +--rw interface?            uint16
   |     |        +--:(segment-identifier)
   |     |        |  +--rw segment-identifier?   fpcbase:segment-id
   |     |        +--:(mpls-label-stack)
   |     |        |  +--rw mpls-label-stack*     fpcbase:fpc-mpls-label
   |     |        +--:(mpls-sr-stack)
   |     |        |  +--rw mpls-sr-stack*        fpcbase:fpc-mpls-label
   |     |        +--:(srv6-stack)
   |     |        |  +--rw srv6-stack*           fpcbase:segment-id
   |     |        +--:(tunnel-info)
   |     |           +--rw tunnel-info
   |     |              +--rw tunnel-local-address?    inet:ip-address
   |     |              +--rw tunnel-remote-address?   inet:ip-address
   |     |              +--rw mtu-size?                uint32
   |     |              +--rw tunnel?                  identityref
   |     |              +--rw payload-type?            enumeration
   |     |              +--rw gre-key?                 uint32
   |     |              +--rw gtp-tunnel-info
   |     |              |  +--rw local-tunnel-identifier?    uint32
   |     |              |  +--rw remote-tunnel-identifier?   uint32
   |     |              |  +--rw sequence-numbers-enabled?   boolean
   |     |              +--rw ebi?                     fpcbase:ebi-type
   |     |              +--rw lbi?                     fpcbase:ebi-type
   |     +--:(qos)
   |        +--rw trafficclass?                  inet:dscp
   |        +--rw per-mn-agg-max-dl?
                    qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value
   |        +--rw per-mn-agg-max-ul?
                    qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value
   |        +--rw per-session-agg-max-dl
   |        |  +--rw max-rate        uint32
   |        |  +--rw service-flag    boolean
   |        |  +--rw exclude-flag    boolean
   |        +--rw per-session-agg-max-ul
   |        |  +--rw max-rate        uint32
   |        |  +--rw service-flag    boolean
   |        |  +--rw exclude-flag    boolean
   |        +--rw priority-level                 uint8
   |        +--rw preemption-capability           enumeration
   |        +--rw preemption-vulnerability        enumeration
   |        +--rw agg-max-dl?
                    qos-pmip:Aggregate-Max-DL-Bit-Rate-Value
   |        +--rw agg-max-ul?
                    qos-pmip:Aggregate-Max-UL-Bit-Rate-Value
   |        +--rw gbr-dl?
                    qos-pmip:Guaranteed-DL-Bit-Rate-Value
   |        +--rw gbr-ul?
                    qos-pmip:Guaranteed-UL-Bit-Rate-Value
   |        +--rw qci?
                    fpcbase:fpc-qos-class-identifier
   |        +--rw ue-agg-max-bitrate?            uint32
   |        +--rw apn-ambr?                      uint32

   policy-configuration-value:
   |  |  |     +--rw (policy-configuration-value)?
   |  |  |        +--:(descriptor-value)
   |  |  |        |  ...
   |  |  |        +--:(action-value)
   |  |  |        |  ...
   |  |  |        +--:(setting-value)
   |  |  |           +--rw setting?                        <anydata>

   policy-configuration:
   |  |  |  +--rw policy-configuration* [index]
   |  |  |     +--rw index                           uint16
   |  |  |     +--rw extensible?             boolean
   |  |  |     +--rw static-attributes*      string
   |  |  |     +--rw mandatory-attributes*   string
   |  |  |     +--rw entity-state?           enumeration
   |  |  |     +--rw version?                uint32
   |  |  |     +--rw (policy-configuration-value)?
   |  |  |        ...

   module: ietf-dmm-fpc
   +--rw tenant* [tenant-key]
     +--rw tenant-key                   fpc:fpc-identity
     +--rw topology-information-model
     |  +--rw service-group* [service-group-key role-key]
     |  |  +--rw service-group-key     fpc:fpc-identity
     |  |  +--rw service-group-name?   string
     |  |  +--rw role-key              identityref
     |  |  +--rw role-name?            string
     |  |  +--rw protocol*             identityref
     |  |  +--rw feature*              identityref
     |  |  +--rw service-group-configuration* [index]
     |  |  |  +--rw index                           uint16
     |  |  |  +--rw (policy-configuration-value)?
     |  |  |        |  ...
     |  |  +--rw dpn* [dpn-key]
     |  |     +--rw dpn-key                 fpc:fpc-identity
     |  |     +--rw referenced-interface* [interface-key]
     |  |        +--rw interface-key            fpc:fpc-identity
     |  |        +--rw peer-service-group-key*   fpc:fpc-identity
     |  +--rw dpn* [dpn-key]
     |  |  +--rw dpn-key                           fpc:fpc-identity
     |  |  +--rw dpn-name?                         string
     |  |  +--rw dpn-resource-mapping-reference?   string
     |  |  +--rw domain-key                        fpc:fpc-identity
     |  |  +--rw service-group-key*                fpc:fpc-identity
     |  |  +--rw interface* [interface-key]
     |  |  |  +--rw interface-key        fpc:fpc-identity
     |  |  |  +--rw interface-name?      string
     |  |  |  +--rw role?                identityref
     |  |  |  +--rw protocol*            identityref
     |  |  |  +--rw interface-configuration* [index]
     |  |  |     +--rw (policy-configuration-value)?
     |  |  |           |  ...
     |  |  +--rw dpn-policy-configuration* [policy-template-key]
     |  |     +--rw policy-template-key     fpc:fpc-identity
     |  |     +--rw policy-configuration* [index]
     |  |        +--rw index    uint16
     |  |          +--rw (policy-configuration-value)?
     |  |               |  ...
     |  +--rw domain* [domain-key]
     |  |  +--rw domain-key        fpc:fpc-identity
     |  |  +--rw domain-name?      string
     |  |  +--rw domain-policy-configuration* [policy-template-key]
     |  |     +--rw policy-template-key     fpc:fpc-identity
     |  |     +--rw policy-configuration* [index]
     |  |           |        ...
     |  +--rw dpn-checkpoint
     |  |  +--rw basename?          fpc:fpc-identity
     |  |  +--rw base-checkpoint?   string
     |  +--rw service-group-checkpoint
     |  |  +--rw basename?          fpc:fpc-identity
     |  |  +--rw base-checkpoint?   string
     |  +--rw dpn-checkpoint
     |  |  +--rw basename?                   fpc:fpc-identity
     |  |  +--rw base-checkpoint?            string
     +--rw policy-information-model
     |  +--rw action-template* [action-template-key]
     |  |  +--rw action-template-key        fpc:fpc-identity
     |  |  +--rw (action-value)
     |  |  |                  ...
     |  |  +--rw extensible?                boolean
     |  |  +--rw static-attributes*         string
     |  |  +--rw mandatory-attributes*      string
     |  |  +--rw entity-state?              enumeration
     |  |  +--rw version?                   uint32
     |  +--rw descriptor-template* [descriptor-template-key]
     |  |  +--rw descriptor-template-key         fpc:fpc-identity
     |  |  +--rw (descriptor-value)
     |  |  |                  ...
     |  |  +--rw extensible?                     boolean
     |  |  +--rw static-attributes*              string
     |  |  +--rw mandatory-attributes*           string
     |  |  +--rw entity-state?                   enumeration
     |  |  +--rw version?                        uint32
     |  +--rw rule-template* [rule-template-key]
     |  |  +--rw rule-template-key           fpc:fpc-identity
     |  |  +--rw descriptor-match-type       enumeration
     |  |  +--rw descriptor-configuration* [descriptor-template-key]
     |  |  |  +--rw descriptor-template-key    fpc:fpc-identity
     |  |  |  +--rw direction?                 rfc5777:direction-type
     |  |  |  +--rw setting?                        <anydata>
     |  |  |  +--rw attribute-expression* [index]
     |  |  |     +--rw index                           uint16
     |  |  |     +--rw (descriptor-value)
     |  |  |     |  ...
     |  |  +--rw action-configuration* [action-order]
     |  |  |  +--rw action-order           uint32
     |  |  |  +--rw action-template-key     fpc:fpc-identity
     |  |  |  +--rw setting?                        <anydata>
     |  |  |  +--rw attribute-expression* [index]
     |  |  |     +--rw index                      uint16
     |  |  |     +--rw (action-value)
     |  |  |     |  ...
     |  |  +--rw extensible?                 boolean
     |  |  +--rw static-attributes*          string
     |  |  +--rw mandatory-attributes*       string
     |  |  +--rw entity-state?               enumeration
     |  |  +--rw version?                    uint32
     |  |  +--rw rule-configuration* [index]
     |  |     +--rw index    uint16
     |  |         +--rw (policy-configuration-value)?
     |  |           |  ...
     |  +--rw policy-template* [policy-template-key]
     |  |  +--rw policy-template-key     fpc:fpc-identity
     |  |  +--rw rule-template* [precedence]
     |  |  |  +--rw precedence          uint32
     |  |  |  +--rw rule-template-key    fpc:fpc-identity
     |  |  +--rw extensible?             boolean
     |  |  +--rw static-attributes*      string
     |  |  +--rw mandatory-attributes*   string
     |  |  +--rw entity-state?           enumeration
     |  |  +--rw version?                uint32
     |  |  +--rw policy-configuration* [index]
     |  |      ...
     |  +--rw basename?              fpc:fpc-identity
     |  +--rw base-checkpoint?       string
     +--rw mobility-context* [mobility-context-key]
     |  +--rw mobility-context-key    fpc:fpc-identity
     |  +--rw delegating-ip-prefix*   inet:ip-prefix
     |  +--rw parent-context?         fpc:fpc-identity
     |  +--rw child-context*          fpc:fpc-identity
     |  +--rw mobile-node
     |  |  +--rw ip-address*    inet:ip-address
     |  |  +--rw imsi?          fpcbase:imsi-type
     |  |  +--rw mn-policy-configuration* [policy-template-key]
     |  |     +--rw policy-template-key     fpc:fpc-identity
     |  |     +--rw policy-configuration* [index]
     |  |        ...
     |  +--rw domain
     |  |  +--rw domain-key?        fpc:fpc-identity
     |  |  +--rw domain-policy-configuration* [policy-template-key]
     |  |     +--rw policy-template-key     fpc:fpc-identity
     |  |     +--rw policy-configuration* [index]
     |  |        ...
     |   +--rw dpn* [dpn-key]
     |      +--rw dpn-key              fpc:fpc-identity
     |      +--rw dpn-policy-configuration* [policy-template-key]
     |      |  +--rw policy-template-key     fpc:fpc-identity
     |      |  +--rw policy-configuration* [index]
     |      |     ...
     |      +--rw role?                identityref
     |      +--rw service-data-flow* [identifier]
     |         +--rw identifier           uint32
     |         +--rw service-group-key?   fpc:fpc-identity
     |         +--rw interface* [interface-key]
     |         |  +--rw interface-key    fpc:fpc-identity
     |         +--rw service-data-flow-policy-
                       configuration* [policy-template-key]
     |            +--rw policy-template-key     fpc:fpc-identity
     |            +--rw policy-configuration* [index]
     |               ...
     +--rw monitor* [monitor-key]
       +--rw extensible?            boolean
       +--rw static-attributes*     string
       +--rw mandatory-attributes*  string
       +--rw entity-state?          enumeration
       +--rw version?               uint32
       +--rw monitor-key            fpc:fpc-identity
       +--rw target?                string
       +--rw deferrable?            boolean
       +--rw (configuration)
          +--:(period)
```

```
           |  +--rw period?           uint32
           +--:(threshold-config)
           |  +--rw low?              uint32
           |  +--rw hi?               uint32
           +--:(schedule)
           |  +--rw schedule?         uint32
           +--:(event-identities)
           |  +--rw event-identities*       identityref
           +--:(event-ids)
              +--rw event-ids*        uint32

      rpcs:
        +---x configure
        |  +---w input
        |  |  +---w client-id          fpc:client-identifier
        |  |  +---w execution-delay?   uint32
        |  |  +---w yang-patch
        |  |     +---w patch-id    string
        |  |     +---w comment?    string
        |  |     +---w edit* [edit-id]
        |  |        +---w edit-id         string
        |  |        +---w operation       enumeration
        |  |        +---w target          target-resource-offset
        |  |        +---w point?          target-resource-offset
        |  |        +---w where?          enumeration
        |  |        +---w value?          <anydata>
        |  |        +---w reference-scope?   fpc:ref-scope
        |  |        +---w command-set
        |  |           +---w (instr-type)?
        |  |              +--:(instr-3gpp-mob)
        |  |              |  +---w instr-3gpp-mob? fpcbase:threegpp-instr
        |  |              +--:(instr-pmip)
        |  |                 +---w instr-pmip?       pmip-commandset
        |  +--ro output
        |     +--ro yang-patch-status
        |        +--ro patch-id       string
        |        +--ro (global-status)?
        |        |  +--:(global-errors)
        |        |  |  +--ro errors
        |        |  |     +--ro error*
        |        |  |        +--ro error-type       enumeration
        |        |  |        +--ro error-tag        string
        |        |  |        +--ro error-app-tag?   string
        |        |  |        +--ro error-path?      instance-identifier
        |        |  |        +--ro error-message?   string
        |        |  |        +--ro error-info?      <anydata>
        |        |  +--:(ok)
        |        |     +--ro ok?            empty
```

```
       |        +--ro edit-status
       |           +--ro edit* [edit-id]
       |              +--ro edit-id            string
       |              +--ro (edit-status-choice)?
       |                 +--:(ok)
       |                 |  +--ro ok?                 empty
       |                 |  +--ro notify-follows?    boolean
       |                 |  +--ro subsequent-edit* [edit-id]
       |                 |     +--ro edit-id      string
       |                 |     +--ro operation    enumeration
       |                 |     +--ro target
                                   ypatch:target-resource-offset
       |                 |     +--ro point?
                                       ypatch:target-resource-offset
       |                 |     +--ro where?       enumeration
       |                 |     +--ro value?       <anydata>
       |                 +--:(errors)
       |                    +--ro errors
       |                       +--ro error*
       |                          +--ro error-type       enumeration
       |                          +--ro error-tag        string
       |                          +--ro error-app-tag?   string
       |                          +--ro error-path?
                                           instance-identifier
       |                          +--ro error-message?   string
       |                          +--ro error-info?      <anydata>
       +---x register_monitor
       |  +---w input
       |  |  +---w client-id          fpc:client-identifier
       |  |  +---w execution-delay?   uint32
       |  |  +---w operation-id              uint64
       |  |  +---w monitor* [monitor-key]
       |  |     +---w extensible?            boolean
       |  |     +---w static-attributes*     string
       |  |     +---w mandatory-attributes*   string
       |  |     +---w entity-state?          enumeration
       |  |     +---w version?               uint32
       |  |     +---w monitor-key            fpc:fpc-identity
       |  |     +---w target?                string
       |  |     +---w deferrable?            boolean
       |  |     +---w (configuration)
       |  |        +--:(period)
       |  |        |  +---w period?                uint32
       |  |        +--:(threshold-config)
       |  |        |  +---w low?                   uint32
       |  |        |  +---w hi?                    uint32
       |  |        +--:(schedule)
       |  |        |  +---w schedule?              uint32
       |  |        +--:(event-identities)
       |  |        |  +---w event-identities*       identityref
       |  |        +--:(event-ids)
       |  |           +---w event-ids*              uint32
       |  +--ro output
       |     +--ro operation-id     uint64
       |     +--ro (edit-status-choice)?
       |        +--:(ok)
       |        |  +--ro ok?       empty
       |        +--:(errors)
       |           +--ro errors
       |              +--ro error*
       |                 +--ro error-type       enumeration
       |                 +--ro error-tag        string
       |                 +--ro error-app-tag?   string
       |                 +--ro error-path?      instance-identifier
       |                 +--ro error-message?   string
       |                 +--ro error-info?      <anydata>
       +---x deregister_monitor
       |  +---w input
       |  |  +---w client-id          fpc:client-identifier
       |  |  +---w execution-delay?   uint32
       |  |  +---w operation-id              uint64
       |  |  +---w monitor* [monitor-key]
       |  |     +---w monitor-key    fpc:fpc-identity
       |  |     +---w send_data?     boolean
       |  +--ro output
       |     +--ro operation-id     uint64
       |     +--ro (edit-status-choice)?
       |        +--:(ok)
       |        |  +--ro ok?       empty
       |        +--:(errors)
       |           +--ro errors
       |              +--ro error*
       |                 +--ro error-type       enumeration
       |                 +--ro error-tag        string
       |                 +--ro error-app-tag?   string
       |                 +--ro error-path?      instance-identifier
       |                 +--ro error-message?   string
       |                 +--ro error-info?      <anydata>
       +---x probe
          +---w input
          |  +---w client-id          fpc:client-identifier
          |  +---w execution-delay?   uint32
          |  +---w operation-id              uint64
          |  +---w monitor* [monitor-key]
          |     +---w monitor-key    fpc:fpc-identity
          +--ro output
```

```
             +--ro operation-id     uint64
             +--ro (edit-status-choice)?
                +--:(ok)
                |  +--ro ok?       empty
                +--:(errors)
                   +--ro errors
                      +--ro error*
                         +--ro error-type       enumeration
                         +--ro error-tag        string
                         +--ro error-app-tag?   string
                         +--ro error-path?      instance-identifier
                         +--ro error-message?   string
                         +--ro error-info?      <anydata>

     notifications:
       +---n config-result-notification
       |  +--ro yang-patch-status
       |  |  +--ro patch-id       string
       |  |  +--ro (global-status)?
       |  |  |  +--:(global-errors)
       |  |  |  |  +--ro errors
       |  |  |  |     +--ro error*
       |  |  |  |        +--ro error-type       enumeration
       |  |  |  |        +--ro error-tag        string
       |  |  |  |        +--ro error-app-tag?   string
       |  |  |  |        +--ro error-path?      instance-identifier
       |  |  |  |        +--ro error-message?   string
       |  |  |  |        +--ro error-info?      <anydata>
       |  |  |  +--:(ok)
       |  |  |     +--ro ok?            empty
       |  |  +--ro edit-status
       |  |     +--ro edit* [edit-id]
       |  |        +--ro edit-id    string
       |  |        +--ro (edit-status-choice)?
       |  |           +--:(ok)
       |  |           |  +--ro ok?        empty
       |  |           +--:(errors)
       |  |              +--ro errors
       |  |                 +--ro error*
       |  |                    +--ro error-type       enumeration
       |  |                    +--ro error-tag        string
       |  |                    +--ro error-app-tag?   string
       |  |                    +--ro error-path?
       |  |                            instance-identifier
       |  |                    +--ro error-message?   string
       |  |                    +--ro error-info?      <anydata>
       |  +--ro subsequent-edit* [edit-id]
       |     +--ro edit-id      string
       |     +--ro operation    enumeration
       |     +--ro target       ypatch:target-resource-offset
       |     +--ro point?       ypatch:target-resource-offset
       |     +--ro where?       enumeration
       |     +--ro value?       <anydata>
       +---n notify
          +--ro notification-id?   uint32
          +--ro timestamp?         uint32
          +--ro report* [monitor-key]
             +--ro monitor-key                fpc:fpc-identity
             +--ro trigger?                   identityref
             +--ro (value)?
                +--:(dpn-candidate-available)
                |  +--ro node-id?                   inet:uri
                |  +--ro supported-interface-list* [role-key]
                |     +--ro role-key    identityref
                +--:(dpn-unavailable)
                |  +--ro dpn-id?                    fpc:fpc-identity
                +--:(report-value)
                   +--ro report-value?              <anydata>
```

Figure 38: YANG FPC Agent Tree

Appendix C. Change Log

C.1. Changes since Version 09

The following changes have been made since version 09

Migration to a Template based framework. This affects all elements. The framework has a template definition language.

Basename is split into two aspects. The first is version which applies to Templates. The second is checkpointing which applies to specific sections only.

Rule was inside Policy and now is Rule-Template and stands as a peer structure to Policy.

Types, e.g. Descriptor Types, Action Types, etc., are now templates that have no values filled in.

The embedded rule has been replaced by a template that has no predefined variables. All rules, pre-configured or embedded, are realized as Policy instantiations.

The Unassigned DPN is used to track requests vs. those that are installed, i.e. Agent assignment of Policy is supported.

The Topology system supports selection information by ServiceGroup or ServiceEndpoint.

DPN Peer Groups and DPN Groups are now PeerServiceGroup and ServiceGroup.

Bulk Configuration and Configuration now follow a style similar to YANG Patch. Agents MAY response back with edits it made to complete the Client edit request.

RFC 5777 Classifiers have been added.

All operations have a common error format.

C.2. Changes since Version 10

The following changes have been made since version 10

Sevice-Endpoints eliminated. Service-Group and DPN interfaces changed to hold information previously held by Service-Endpoint as noted in ML during IETF 101.

Service-Group resides under the Topology-Information-Mode

The Domain now has a checkpoint and the Topology Information Model checkpoint was removed to avoid any overlaps in checkpoints.

Scrubbed YANG for NMDA compliance and Guidelines (RFC 6087bis).

Monitor lifecycle, policy and policy installation examples added.

Authors’ Addresses

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

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

Lyle Bertz
6220 Sprint Parkway
Overland Park KS, 66251
USA

Email: lylebe551144@gmail.com

Marco Liebsch
NEC Laboratories Europe
NEC Europe Ltd.
Kurfuersten-Anlage 36
D-69115 Heidelberg
Germany

Phone: +49 6221 4342146
Email: liebsch@neclab.eu

Sri Gundavelli
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sgundave@cisco.com

Danny Moses

Email: danny.moses@intel.com

Charles E. Perkins
Futurewei Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1-408-330-4586
Email: charliep@computer.org

DMM Working Group
Internet-Draft
Intended status: Informational
Expires: January 31, 2018

A. Yegin
Actility
D. Moses
Intel
K. Kweon
J. Lee
J. Park
Samsung
S. Jeon
Sungkyunkwan University

July 30, 2017

On Demand Mobility Management
draft-ietf-dmm-ondemand-mobility-12

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 by selectively providing IP session continuity and IP address reachability on a per-socket basis.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In the context of Mobile IP [RFC5563][RFC6275][RFC5213][RFC5944], the following two attributes are defined for IP service provided to 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 changing 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 its 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 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 IP session continuity and IP address reachability.

It should be noted that in reality not every application may need these 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., ping) can survive without having IP session continuity support.

Achieving IP session continuity and IP address reachability with 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 [RFC7333]. Therefore, IP session continuity and IP address reachability should be provided only when necessary.

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. These 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, the higher-layer protocols are rendered useless because their operation is inhibited by Mobile IP. Since Mobile IP ensures that 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 applications running on mobile hosts 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. 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

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

- Fixed IP Address

A Fixed IP address is an address with a guarantee to be valid for a very long time, regardless of whether it is being used in any packet to/from the mobile host, or whether or not the mobile host is connected to the network, or whether it moves from one point-of-attachment to another (with a different IP prefix) while it is connected.

Fixed IP addresses are required by applications that need both IP session continuity and IP address reachability.

- Session-lasting IP Address

A session-lasting IP address is an address with a guarantee to be valid throughout the IP session(s) for which it was requested. It is guaranteed to be valid even after the mobile host had moved from one point-of-attachment to another (with a different IP prefix).

Session-lasting IP addresses are required by applications that need IP session continuity but do not need IP address reachability.

- Non-persistent IP Address

This type of IP address does not provide IP session continuity nor IP address reachability. The IP address is created from an IP prefix that is obtained from the serving IP gateway and is not maintained across gateway changes. In other words, the IP prefix may be released and replaced by a new one when the IP gateway changes due to the movement of the mobile host forcing the creation of a new source IP address with the updated allocated IP prefix.

- Graceful Replacement IP Address

In some cases, the network cannot guarantee the validity of the provided IP prefix throughout the duration of the IP session, but can provide a limited graceful period of time in which both the original IP prefix and a new one are valid. This enables the application some flexibility in the transition from the existing source IP address to the new one.

This gracefulness is still better than the non-persistence type of address for applications that can handle a change in their source IP address but require that extra flexibility.

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. 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 Session-lasting IP Addresses. For example: Web browsers.

Applications with very short IP sessions, such as DNS clients and instant messengers, can utilize Non-persistent IP Addresses. Even though they could very well use Fixed or Session-lasting IP Addresses, the transmission latency would be minimized when a Non-persistent IP Addresses are used.

Applications that can tolerate a short interruption in connectivity can use the Graceful-replacement IP addresses. For example, a streaming client that has buffering capabilities.

3.2. Granularity of Selection

IP address type selection is made on a per-socket granularity. Different parts of the same application may have different needs. For example, the control-plane of an application may require a Fixed

IP Address in order to stay reachable, whereas the data-plane of the same application may be satisfied with a Session-lasting 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 Non-persistent, zero or more Session-lasting, zero or more Fixed and zero or more Graceful-Replacement IP addresses may be configured by 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 an application requires a specific type of IP address and such an address is not already configured on the host, the IP stack shall attempt to configure one. For example, a host may not always have a Session-lasting IP address available. When an application requests one, the IP stack shall make an attempt to configure one by issuing a request to the network (see Section 3.4 below for more details). If the operation fails, the IP stack shall fail the associated socket request and return an error. If successful, a Session-lasting IP Address gets configured on the mobile host. If another socket requests a Session-lasting IP address at a later time, the same IP address may be served to that socket as well. When the last socket using the same configured IP address is closed, the IP address may be released or kept for future applications that may be launched and require a Session-lasting IP address.

In some cases it might be preferable for the mobile host to request a new Session-lasting IP address for a new opening of an IP session (even though one was already assigned to the mobile host by the network and might be in use in a different, already active IP session). It is outside the scope of this specification to define criteria for choosing to use available addresses or choosing to request new ones. It supports both alternatives (and any combination).

It is outside the scope of this specification to define how the host requests a specific type of prefix and how the network indicates the type of prefix in its advertisement or in its reply to a request).

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 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 Desired Address Type

[RFC5014] introduced the ability of applications to influence the source address selection with the IPV6_ADDR_PREFERENCE option at the IPPROTO_IPV6 level. This option is used with setsockopt() and getsockopt() calls to set/get address selection preferences.

Extending this further by adding more flags does not work when a request for an address of a certain type results in requiring the IP stack to wait for the network to provide the desired source IP prefix and hence causing the setsockopt() call to block until the prefix is allocated (or an error indication from the network is received).

Alternatively a new Socket API is defined - getsc() which allows applications to express their desired type of session continuity service. The new getsc() API will return an IPv6 address that is associated with the desired session continuity service and with status information indicating whether or not the desired service was provided.

An application that wishes to secure a desired service will call getsc() with the service type definition and a place to contain the provided IP address, and call bind() to associate that IP address with the Socket (See pseudo-code example in Section 4 below).

When the IP stack is required to use a source IP address of a specified type, it can use an existing address, or request a new IP prefix (of the same type) from the network and create a new one. If the host does not already have an IPv6 prefix of that specific type, it must request one from the network.

Using an existing address from an existing prefix is faster but might yield a less optimal route (if a hand-off event occurred after its configuration). On the other hand, acquiring a new IP prefix from the network may be slower due to signaling exchange with the network.

Applications can control the stack's operation by setting a new flag - ON_NET flag - which directs the IP stack whether to use a preconfigured source IP address (if exists) or to request a new IPv6 prefix from the current serving network and configure a new IP address.

This new flag is added to the set of flags in the IPV6_ADDR_PREFERENCES option at the IPPROTO_IPV6 level. It is used in setsockopt() to set the desired behavior.

4. Usage example

The following example shows pseudo-code for creating a Stream socket (TCP) with a Session-Lasting source IP address:

```
#include <sys/socket.h>
#include <netinnet/in.h>

  // Socket information
int              s ;               // Socket id

  // Source information (for secsc() and bind())
sockaddr_in6     sourceInfo         // my address and port for bind()
in6_addr         sourceAddress      // will contain the provisioned source
                                    // IP address
uint8_t          sc_type = IPV6_REQUIRE_SESSION_LASTING_IP ;
                                    // For requesting a Session-Lasting
                                    // source IP address

  // Destination information (for connect())
sockaddr_in6     serverInfo ;       // server info for connect()

  // Create an IPv6 TCP socket
s = socket(AF_INET6, SOCK_STREAM, 0) ;
if (s!=0) {
      // Handle socket creation error
      // ...
} // if socket creation failed
else {
       // Socket creation is successful
       // The application cannot connect yet, since it wants to use a
       // Session-Lasting source IP address It needs to request the
       // Session-Lasting source IP before connecting
     if (setsc(s, &sourceAddress, &sc_type)) == 0){
          // setting session continuity to Session Lasting is successful
          // sourceAddress now contains the Session-Lasting source IP
          // address

          // Bind to that source IP address
        sourceInfo.sin6_family = AF_INET6 ;
        sourceInfo.sin6_port = 0    // let the stack choose the port
        sourceInfo.sin6_address = sourceAddress ;
                                    // Use the source address that was
                                    // generated by the setsc() call
        if (bind(s, &sourceInfo, sizeof(sourceInfo))==0){
             // Set the desired server’s information for connect()
           serverInfo.sin6_family = AF_INET6 ;
           serverInfo.sin6_port = SERVER_PORT_NUM ;
           serverAddress.sin6_addr = SERVER_IPV6_ADDRESS ;

             // Connect to the server
           if (connect(s, &serverInfo, sizeof(serverInfo))==0) {
               // connect successful (3-way handshake has been completed
               // with Session-Lasting source address.
               // Continue application functionality
               // ...
           }  // if connect() is successful
           else {
               // connect failed
               // ...
               // Application code that handles connect failure and closes
               // the socket
               // ...
           } // if connect() failed
        } // if bind() successful
        else {
               // bind() failed
               // ...
               // Application code that handles bind failure and closes
               // the socket
               // ...
            } // if bind() failed
     }  // if setsc() was successful and of a Session-Lasting source address was
 provided
     else {
          // application code that does not use Session-lasting IP address
          // The application may either connect without the desired
          // Session-lasting service, or close the socket
          //...
     } // if setsc() failed
}  // if socket was created successfully

  // The rest of the application’s code
  // ...
```

5. 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

5.1. Applications

Legacy applications that do not support the OnDemand functionality will use the legacy API and will not be able to take advantage of the On-Demand Mobility feature.

Applications using the new OnDemand functionality must be aware that they may be executed in legacy environments that do not support it. Such environments may include a legacy IP stack on the mobile host, legacy network infrastructure, or both. In either case, the API will return an error code and the invoking applications may just give up and use legacy calls.

5.2. IP Stack in the Mobile Host

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

If the network infrastructure supports On-Demand functionality, the IP stack should follow the application request: If the application requests a specific address type, the stack should forward this request to the network. If the application does not request an address type, the IP stack must not request an address type and leave it to the network’s default behavior to choose the type of the allocated IP prefix. If an IP prefix was already allocated to the host, the IP stack uses it and may not request a new one from the network.

5.3. Network Infrastructure

The network infrastructure may or may not support the On-Demand functionality. 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.

5.4. Merging this work with RFC5014

[RFC5014] defines new flags that may be used with setsockopt() to influence source IP address selection for a socket. The list of flags include: source home address, care-of address, temporary address, public address CGA (Cryptographically Created Address) and non-CGA. When applications require session continuity service and use setsc() and bind(), they should not set the flags specified in [RFC5014].

However, if an application sets a specific option using setsockopt() with one of the flags specified in [RFC5014] and also selects a source IP address using setsc() and bind() the IP address that was generated by setsc() and bound using bind() will be the one used by traffic generated using that socket and options set by setsockopt() will be ignored.

If bind() was not invoked after setsc() by the application, the IP address generated by setsc() will not be used and traffic generated by the socket will use a source IP address that complies with the options selected by setsockopt().

6. Summary of New Definitions

6.1. New APIs

setsc() enables applications to request a specific type of source IP address in terms of session continuity. Its definition is:

```
int setsc (int sockfd, in6_addr *sourceAddress, sc_type addressType) ;

Where:
 - sockfd -       is the socket descriptor of the socket with which a
                  specific address type is associated
 - sourceAddress - is a pointer to an area allocated for setsc() to place
                  the generated source IP address of the desired session
                                  continuity type
 - addressType -   Is the desired type of session continuity service.
                  It is a 3-bit field containing one of the following
                                  values:
                  0 - Reserved
                  1 - FIXED_IPV6_ADDRESS
                  2 - SESSION_LASTING_IPV6_ADDRESS
                  3 - NON_PERSISTENT_IPV6_ADDRESS
                  4 - GRACEFUL_REPLACEMENT_IPV6_ADDRESS
                  5-7 - Reserved

setsc() returns the status of the operation:
 - 0 - Address was successfully generated
 - EAI_REQUIREDIPNOTSUPPORTED - the required service type is not supported
 - EAI_REQUIREDIPFAILED - the network could not fulfill the desired request
```

setsc() may block the invoking thread if it triggers the TCP/IP stack to request a new IP prefix from the network to construct the desired source IP address. If an IP prefix with the desired session continuity features already exists (was previously allocated to the mobile host) and the stack is not required to request a new one as a result of setting the IPV6_REQUIRE_SRC_ON_NET flag (defined below), setsc() may return immediately with the constructed IP address and will not block the thread.

6.2. New Flags

The following flag is added to the list of flags in the IPV6_ADDR_PREFERENCE option at the IPPROTO6 level:

IPV6_REQUIRE_SRC_ON_NET - set IP stack address allocation behavior

If set, the IP stack will request a new IPv6 prefix of the desired type from the current serving network and configure a new source IP address. If reset, the IP stack will use a preconfigured one if it exists. If there is no preconfigured IP address of the desired type, a new prefix will be requested and used for creating the IP address.

7. 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.

8. IANA Considerations

This document has no IANA considerations.

9. Contributors

This document was merged with [I-D.sijeon-dmm-use-cases-api-source]. We would like to acknowledge the contribution of the following people to that document as well:

Sergio Figueiredo
Altran Research, France
Email: sergio.figueiredo@altran.com

Younghan Kim
Soongsil University, Korea
Email: younghak@ssu.ac.kr

John Kaippallimalil
Huawei, USA
Email: john.kaippallimalil@huawei.com

10. Acknowledgements

We would like to thank Wu-chi Feng, Alexandru Petrescu, Jouni Korhonen, Sri Gundavelli, Dave Dolson and Lorenzo Colitti for their valuable comments and suggestions on this work.

11. References

11.1. 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>.

[RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, DOI 10.17487/RFC5014, September 2007, <http://www.rfc-editor.org/info/rfc5014>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

11.2. Informative References

[I-D.sijeon-dmm-use-cases-api-source] Jeon, S., Figueiredo, S., Kim, Y., and J. Kaippallimalil, "Use Cases and API Extension for Source IP Address Selection", draft-sijeon-dmm-use-cases-api-source-06 (work in progress), March 2017.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.

[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>.

[RFC5563] Leung, K., Dommety, G., Yegani, P., and K. Chowdhury, "WiMAX Forum / 3GPP2 Proxy Mobile IPv4", RFC 5563, DOI 10.17487/RFC5563, February 2010, <http://www.rfc-editor.org/info/rfc5563>.

[RFC5944] Perkins, C., Ed., "IP Mobility Support for IPv4, Revised", RFC 5944, DOI 10.17487/RFC5944, November 2010, <http://www.rfc-editor.org/info/rfc5944>.

[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>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[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>.

Authors' Addresses

Alper Yegin
Actility
Istanbul
Turkey

Email: alper.yegin@actility.com

Danny Moses
Intel Corporation
Petah Tikva
Israel

Email: danny.moses@intel.com

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

Seil Jeon
Sungkyunkwan University
Suwon
South Korea

Email: seiljeon@skku.edu

SPRING and DMM   S. Matsushima
Internet-Draft   SoftBank
Intended status: Standards Track   C. Filsfils
Expires: January 18, 2018   Cisco Systems, Inc.
                                                           July 17, 2017

SRv6 for Mobile User-Plane
draft-matsushima-spring-dmm-srv6-mobile-uplane-01

Abstract

This document discusses the applicability of SRv6 (Segment Routing IPv6) to user-plane of mobile networks that SRv6 source routing capability with its programmability can fulfill the user-plane functions, such as access and anchor functions. It takes advantage of underlying layer awareness and flexibility to deploy user-plane functions that enables optimizing data-path for applications. Network slicing and an interworking way between SRv6 and existing mobile user-plane are also discussed in this document.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In mobile networks, mobility management systems provide connectivity while mobile nodes move around. While the control-plane of the system signals movements of a mobile node, user-plane establishes tunnel between the mobile node and anchor node over IP based backhaul and core networks.

This document discusses the applicability of SRv6 (Segment Routing IPv6) to those mobile networks. SRv6 provides source routing to networks where operators can explicitly indicate route for the packets from and to the mobile node. SRv6 endpoint nodes act as roles of anchor of mobile user-plane.

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 segment routing and SRv6 network programming terms are defined in [I-D.ietf-spring-segment-routing] and "[I-D.filsfils-spring-srv6-network-programming].

3. Motivations

Today’s and future applications are requiring highly optimized data-path between mobile nodes and the entities of those applications in perspectives of latency, bandwidth, etc,. However current architecture of mobile management is agnostic about underlying topologies of transport layer. It rigidly fragments the user-plane in radio access, core and service networks and connects them by tunneling techniques through the user-plane functions such as access and anchor nodes. Those agnostic and rigidness make it difficult for the operator to optimize the data-path.

While the mobile network industry has been trying to solve that, applications shift to use IPv6 data-path and network operators adopt it as their IP transport as well. SRv6 integrates both application data-path and underlying transport layer in data-path optimization aspects that does not require any other techniques.

SRv6 source routing capability with programmable functions [I-D.filsfils-spring-srv6-network-programming] could fulfills the user-plane functions of mobility management. It takes advantage of underlying layer awareness and flexibility to deploy user-plane functions. Those are the motivations to adopt SRv6 for mobile user-plane.

4. Mobile User-Plane

This section describes user-plane using SRv6 for mobile networks. This clarifies mobile user-plane functions to which SRv6 endpoint applied.

Figure 1 shows mobile user-plane functions which are connected through SRv6 enabled IPv6 networks. In the Figure 1, an mobile node (MN) connects to an SRv6 endpoint serving access node role for the MN. When the endpoint receives packets from the MN, it pushes SRH to the packets. The segment list in the SRH indicates the rest of user-plane segments which are L2 and L3 anchors respectively. Then the endpoint send the packets to the SRv6 enabled IPv6 network. In opposite direction, when an SRv6 endpoint serving L3 anchor role for the MN receives packets to it, the endpoint push SRH consist of the L2 anchor and access node segments to the packets.

```
                                User-plan
                                Function
                               <L2 Anchor>
                                O------O
                                | SRv6 |
                                | End  |
                                | Point|
                                O------O
              User-plan            ||            User-plan
    [MN]      Function        _____||_____      Function
      |     <Access Node>    /            \    <L3 Anchor>
   ___v___     O------O     /              \     O------O     ________
  /  Radio \   | SRv6 |    /      SRv6      \    | SRv6 |    /        \
 /  Access  \==| End  |===/      enabled     \===| End  |===/ Service  \
 \    NW    /  | Point|   \       IPv6       /   | Point|   \    NW    /
  \________/   O------O    \       NW       /    O------O    \________/
                            \              /
                             \____________/
```

Figure 1: Mobile User-plane with SRv6

An SRv6 segment represents those each function, such as Access Node, Layer-2 (L2) Anchor and Layer-3 (L3) Anchor. This makes mobile networks highly flexible to deploy any user-plane functions to which nodes in user flow basis. An SRv6 segment can represent a set of flows in any granularity of aggregation even though it is just for a single flow.

Figure 2 shows that an SRv6 endpoint connects existing IPv4 mobile user-plane, which is defined in [RFC5213] and [TS.29281]. An SRv6 segment in the endpoint represents interworking function which enables interworking between existing access node and SRv6 anchor segment, or SRv6 access node segment and existing anchor node.

Existing mobile user-plane with IPv6 underlay is expected to be widely deployed. As IPv6 network should be interoperable with SRv6 enabled network and SRv6 endpoints can be accommodated on it, interworking with existing IPv6 network is out of scope of this document.

```
     ________                                                 _______
    /        \    O------O                                   /       \
   / Service  \===|L2/L3 |                                  / Service \
   \    NW    /   |Anchor|         User-plan                \    NW   /
    \________/    |Node  |         Function                  \_______/
                  O------O       <Interworking>                  ||
                      \\________    O------O     _________    O------O
                      /        \    | SRv6 |    /  SRv6  \    | SRv6 |
                     / Existing \===| End  |===/  enabled \===| End  |
                     \  IPv4 NW /   | Point|   \  IPv6 NW /   | Point|
       [MN]           \_________/   O------O    \_________/   O------O
        |             //                                         ||
     ___v____    O------O                                     ___||__
    / Radio  \   |      |                                    / Radio  \
   /  Access  \==|Access|                            [MN]~~~/  Access  \
   \    NW    /  |Node  |                                   \    NW    /
    \________/   O------O                                    \_________/
```

Figure 2: Interworking with Existing Mobile Networks

The detail of SRv6 segments representing user-plane functions are described in Section 5.

5. User-Plane Functions

This section describes mobile user-plane functions applied to SRv6 segment. SRv6 endpoint is capable to do that. Terminology of endpoint functions refers to the net-pgm draft. SRv6 endpoint functions are considered for each direction, such as uplink (UL) from mobile node to the correspondent node, and downlink (DL) from the correspondent node to mobile node.

5.1. Access Node Segment

Access node segment provides SRv6 endpoint the role to which mobile node is connected directly. eNodeB could be referenced as an entity implementing the access node in 3GPP term. The applicable SRv6 functions for the access nodes are following:

UL: T.Insert, or T.Encaps

DL: End.X, or End.DX2/4

In uplink, an endpoint of an access node segment does T.Insert process for the receiving packets from mobile node when the packets matched to the policy bound to the segment in case IPv6 is the outer header of the packet. When the header is IPv4, or other types, the endpoint does T.Encap process for the packets. The segment list represents policy of data-path for the flows from mobile node.

In downlink, the endpoint does End.X process for receiving packets when the active segment represents access node to the mobile node. When the segment left of receiving packets is zero, and IPv4 is indicated as the effective next header (ENH), the endpoint does End.DX4 process for the packets. When the ENH indicates layer-2 type, the endpoint does End.DX2 process for it.

5.2. Layer-2 Anchor Segment

Layer-2 anchor segment provides SRv6 endpoint the role to be anchor point while mobile node move around within a serving area which could be assumed as a layer-2 network. Serving Gateway (SGW) could be referenced as an entity implementing the layer-2 anchor in 3GPP term. The applicable SRv6 functions for the layer-2 anchor are following:

UL: End, or End.B6

DL: Same as UL

In uplink, an endpoint of a L2 anchor segment does End process for the packets when the active segment represents the L2 anchor. When the segment is bound to another policy of data-path, the endpoint does End.B6 process for the packets to be inserted a SRH which consists of segment list along with the policy. The last segment of the inserted SRH must be the edge endpoint of the SRv6 domain of mobile network.

In downlink, there’s no difference between the uplink behavior.

5.3. Layer-3 Anchor Segment

Layer-3 anchor segment provides SRv6 endpoint the role to be anchor point across a mobile network consists of multiple serving areas. Packet data network gateway (PGW) could be referenced as an entity implementing the layer-3 anchor. The applicable SRv6 functions for the layer-3 anchor are following:

UL: End.T, End.DT4, or End.DX2

DL: T.Insert, or T.Encaps

In uplink, an endpoint of a L3 anchor segment does End.T process for the packets when the active segment represents the L3 anchor and specific IPv6 routing table is bound to it. Those routing tables may be accommodated in the endpoint for networks which require individual routing policies. When the segment left of receiving packet is zero and IPv4 is indicated as the effective next header (ENH), the endpoint does End.DT4 process for the packets with specific IPv4 routing table bound to the segment. When the ENH indicates layer-2 type, the endpoint does End.DX2 process for it.

In downlink, an endpoint of a L3 anchor segment does T.Insert process for the packets when the IPv6 flow of packets matched to the policy bound to the segment. When the matched flow is IPv4, or other types, the endpoint does T.Encap process for the packets. The segment list represents policy of data-path for the flows.

5.4. Stateless Interworking Segment

Interworking segment provides SRv6 endpoint the role to be interworking point between existing mobile user-plane and SRv6 mobile user-plane. It is expected that the endpoint of interworking segment could be unaware from the control-plane of the mobility management. While there are combinations of interworking either existing or SRv6 network in which user-plane functions accommodate, interworking segment should cover all combinations without mobility state.

To fulfill the above requirements, SID for interworking segment is encoding identifiers of corresponding tunnel in existing network as argument of the SID. Tunnel encoding format in SID is following:

```
+----------------------+------+-------+-------+
|Locater of interwork  |  DA  |  SA   | Tun-ID|
+----------------------+------+-------+-------+
        128-a-b-c          a      b       c
```

Figure 3: Stateless Interworking SID Encoding

In SRv6 to existing network direction, an endpoint of an interworking segment allocate a node local SID prefix to interworking segments. When the endpoint receives packet and the active segment of the packet indicates the SID, the endpoint pop the SRH of the SID, and then the endpoint encaps the payload with the encoded information in the SID which are tunnel identifier of tunnel header, source and destination IPv4 address of IPv4 header described in Figure 3. Then the endpoint send out the packet to the existing network along with its routing policy.

In existing network to SRv6 network direction, existing network allocates IPv4 address spaces routed to interworking SRv6 network. SRv6 network allocates a domain-wise SID prefix for interworking segments. When a SRv6 endpoint connects to existing network receives packet destined to the allocated IPv4 address, the endpoint decaps outer IPv4 and tunnel header. And then the endpoint does T.Insert process with the SID which consists of the allocated domain-wise SID prefix, source and destination addresses, and tunnel identifier as described in Figure 3. Then the endpoint send out the packet to the SRv6 network along with its routing policy.

In case of IPv4 flow packet over the user-plane, the endpoint does IPv6 header encaps and decaps instead of SRH insert and pop process. The IPv6 header includes interworking segment SID in the SRH.

Noted that to make sure stateless interworking, entities of control-plane in mobile management should cooperate with SRv6 user-plane settings. Further control-plane consideration is discussed in Section 7.

5.5. Rate Limit Segment

Mobile user-plane requires rate-limit feature. SID is able to encode limiting rate as an argument in SID. Multiple flows of packets should have same group identifier in SID when those flows are in an same AMBR group. This helps to keep user-plane stateless. That enables SRv6 endpoint nodes which are unaware from the mobile control-plane information. Encoding format of rate limit segment SID is following:

```
           +----------------------+----------+-----------+
           | Locater of rate-limit| group-id | limit-rate|
           +----------------------+----------+-----------+
                     128-i-j           i          j
```

Figure 4: Stateless Interworking SID Encoding

6. Network Slicing Considerations

Mobile network may be required to create a network slicing that represent a set of network resources and isolate those resource from other slices. User-plane functions represented as SRv6 segments would be part of a slice.

To represent a set of user-plane function segments for a slice, sharing same prefix through those SIDs within the slice could be a straightforward way. SIDs in a network slice may include other type of functions in addition to the mobile user-plane functions described in this document, and underlay integration to meet SLA and quality requirements.

While network slicing has been discussed in the IETF and other standardization bodies, what functionalities are required for network slicing in mobile user-plane is further study item and to be discussed.

7. Control Plane Considerations

Mobile control-plane entities must allocate SIDs to user-plane function segments in case of those entities are distributed to accommodate in the SRv6 endpoints, or those are separated from the endpoint but each of them corresponds to each SRv6 endpoint. In latter case, control-plane entity must advertise allocated SID to the endpoint through some means.

When a centralized controller interfaces to mobile control-planes is capable to allocate SIDs to the controlling SRv6 endpoints, the mobile control-planes just need to indicate the endpoint nodes and their user-plane roles to the controller. In this case, the controller must allocate appropriate SIDs for the user-plane roles to the indicated SRv6 endpoints. The controller must advertise allocated SIDs to the endpoints.

To indicate endpoints and their user-plane functions from mobile control-plane to user-plane, the endpoint or the controller could take advantage of [I-D.ietf-dmm-fpc-cpdp]. It provides interface to the control-plane to manage the user-plane of mobile networks.

In case of stateless interworking, SID allocating entity needs to be aware SID prefix which interworking SRv6 endpoint and SRv6 domain allocate discussed in Section 5.4. The mobile control-plane also need to allocate tunnel endpoint IPv4 address to which corresponding interworking segment destined from existing user-plane that is also discussed in Section 5.4.

8. Security Considerations

TBD

9. IANA Considerations

This document has no actions for IANA.

10. References

10.1. Normative References

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Matsushima, S., Lebrun, D., Decraene, B., Peirens, B., Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P., Sharif, M., Ayyangar, A., Mynam, S., Henderickx, W., Bashandy, A., Raza, K., Dukes, D., Clad, F., and P. Camarillo, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-01 (work in progress), June 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-12 (work in progress), June 2017.

[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>.

10.2. Informative References

[I-D.ietf-dmm-fpc-cpdp]
Matsushima, S., Bertz, L., Liebsch, M., Gundavelli, S., Moses, D., and C. Perkins, "Protocol for Forwarding Policy Configuration (FPC) in DMM", draft-ietf-dmm-fpc-cpdp-07 (work in progress), March 2017.

[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>.

[TS.29281]
3GPP, , "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", 3GPP TS 29.281 10.3.0, September 2011.

Authors’ Addresses

Satoru Matsushima
SoftBank
Tokyo
Japan

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

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

SPRING and DMM
Internet-Draft
Intended status: Standards Track
Expires: May 16, 2018

S. Matsushima
SoftBank
C. Filsfils
M. Kohno
Cisco Systems, Inc.
D. Voyer
Bell Canada

November 12, 2017

Segment Routing IPv6 for Mobile User-Plane
draft-matsushima-spring-dmm-srv6-mobile-uplane-03

Abstract

This document discusses the applicability of SRv6 (Segment Routing IPv6) to user-plane of mobile networks that SRv6 source routing capability with its programmability can fulfill the user-plane functions, such as access and anchor functions. It takes advantage of underlying layer awareness and flexibility to deploy user-plane functions that enables optimizing data-path for applications. Network slicing and an interworking way between SRv6 and existing mobile user-plane are also discussed in this document.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In mobile networks, mobility management systems provide connectivity while mobile nodes move around. While the control-plane of the system signals movements of a mobile node, user-plane establishes tunnel between the mobile node and anchor node over IP based backhaul and core networks.

This document discusses the applicability of SRv6 (Segment Routing IPv6) to those mobile networks. SRv6 provides source routing to networks where operators can explicitly indicate route for the packets from and to the mobile node. SRv6 endpoint nodes act as roles of anchor of mobile user-plane.

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 segment routing and SRv6 network programming terms are defined in [I-D.ietf-spring-segment-routing] and "[I-D.filsfils-spring-srv6-network-programming].

3. Motivations

Today’s and future applications are requiring highly optimized data-path between mobile nodes and the entities of those applications in perspectives of latency, bandwidth, etc,. However current architecture of mobile management is agnostic about underlying topologies of transport layer. It rigidly fragments the user-plane in radio access, core and service networks and connects them by tunneling techniques through the user-plane functions such as access and anchor nodes. Those agnostic and rigidness make it difficult for the operator to optimize the data-path.

While the mobile network industry has been trying to solve that, applications shift to use IPv6 data-path and network operators adopt it as their IP transport as well. SRv6 integrates both application data-path and underlying transport layer in data-path optimization aspects that does not require any other techniques.

SRv6 source routing capability with programmable functions [I-D.filsfils-spring-srv6-network-programming] could fulfills the user-plane functions of mobility management. It takes advantage of underlying layer awareness and flexibility to deploy user-plane functions. Those are the motivations to adopt SRv6 for mobile user-plane.

4. Mobile User-Plane

This section describes user-plane using SRv6 for mobile networks. This clarifies mobile user-plane functions to which SRv6 endpoint applied.

Figure 1 shows mobile user-plane functions which are connected through IPv6-only networks. In the Figure 1, an mobile node (MN) connects to an SRv6 endpoint serving access point role for the MN. When the endpoint receives packets from the MN, it pushes SRH to the packets. The segment list in the SRH indicates the rest of user-plane segments which are L2 and L3 anchors respectively. Then the endpoint send the packets to the IPv6 network. In opposite direction, when an SRv6 endpoint serving L3 anchor role for the MN receives packets to it, the endpoint push SRH consist of the L2 anchor and access point segments to the packets.

```
                                User-plane
                                Function
                               <L2 Anchor>
                                O------O
                                | SRv6 |
                                | End  |
                                | Point|
                                O------O
              User-plane           ||           User-plane
    [MN]      Function        _____||_____      Function
      |     <Access Point>   /            \    <L3 Anchor>
   ___v___     O------O     /              \     O------O     ________
  /  Radio \   | SRv6 |    /                \    | SRv6 |    /        \
 /  Access  \==| End  |===/     IPv6-Only    \===| End  |===/ Service  \
 \    NW    /  | Point|   \      Network     /   | Point|   \    NW    /
  \________/   O------O    \                /    O------O    \________/
                            \              /
                             \____________/
```

Figure 1: Mobile User-plane with SRv6

An SRv6 segment represents those each function, such as Access Point, Layer-2 (L2) Anchor and Layer-3 (L3) Anchor. This makes mobile networks highly flexible to deploy any user-plane functions to which nodes in user flow basis. An SRv6 segment can represent a set of flows in any granularity of aggregation even though it is just for a single flow.

Figure 2 shows that an SRv6 endpoint connects existing IPv4 mobile user-plane, which is defined in [RFC5213] and [TS.29281]. An SRv6 segment in the endpoint represents interworking function which enables interworking between existing access point and SRv6 anchor segment, or SRv6 access point segment and existing anchor node.

Existing mobile user-plane with IPv6 underlay is expected to be widely deployed. As IPv6 network should be interoperable with SRv6 endpoints can be accommodated on it, interworking with existing IPv6 network is out of scope of this document.

```
     ________                                                 _______
    /        \    O------O                                   /       \
   / Service  \===|L2/L3 |                                  / Service \
   \    NW    /   |Anchor|         User-plane               \    NW   /
    \________/    |Node  |         Function                  \_______/
                  O------O       <Interworking>                  ||
                      \\_______     O------O     ________     O------O
                      /        \    | SRv6 |    /        \    | SRv6 |
                     / Existing \===| End  |===/ IPv6-Only\===| End  |
                     \  IPv4 NW /   | Point|   \  Network /   | Point|
       [MN]           \________/    O------O    \________/    O------O
        |             //                                         ||
     ___v____    O------O                                     ___||__
    / Radio  \   |Access|                                    / Radio  \
   /  Access  \==|Point |                            [MN]~~~/  Access  \
   \    NW    /  |Node  |                                   \    NW    /
    \________/   O------O                                    \________/
```

Figure 2: Interworking with Existing Mobile Networks

The detail of SRv6 segments representing user-plane functions are described in Section 5.

5. Supporting Mobile User-Plane Functions

This section describes mobile user-plane functions to which an SRv6 node can apply SRv6 functions and behaviors. The SRv6 node configured with those segments thereby fulfills the user-plane functions. Each function consist of two segments which are uplink (UL) from mobile node to the correspondent node, and downlink (DL) from the correspondent node to mobile node.

An SRv6 node may be configured with multiple type of user-plane functions. Each function may also be configured with multiple sets of the segments for one type of function that to purpose of separating tenants, resources and service policies, etc.

5.1. Access Point

Access Point function provides SRv6 node the role to which mobile node is connected directly. eNodeB could be referenced as an entity implementing the access point in 3GPP term.

When an SRv6 node is configured for an Access Point function, the SRv6 node allocates one DL access-point segment SID per session, or per Access Point function which represents one policy that is shared by multiple sessions.

Applicable SRv6 functions and behaviors are determined by use cases described in Section 6.

5.2. Layer-2 Anchor

Layer-2 anchor function provides SRv6 node the role to be anchor point while mobile node move around within a serving area which could be assumed as a layer-2 network. Serving Gateway (SGW) could be referenced as an entity implementing the layer-2 anchor in 3GPP term.

When an SRv6 node is configured for a Layer-2 anchor function, the SRv6 node allocates UL L2-anchor segment SID per SRv6 policy, which is bound to next L3-anchor function and specific service if needed. The SRv6 node also allocates one DL L2-anchor segment SID per SRv6 policy, which is bound to serving access point SID and specific service if needed.

Applicable SRv6 functions and behaviors are determined by use cases described in Section 6.

5.3. Layer-3 Anchor

Layer-3 anchor function provides SRv6 node the role to be anchor point across a mobile network consists of multiple serving areas. Packet data network gateway (PGW) could be referenced as an entity implementing the layer-3 anchor.

When an SRv6 node is configured for a Layer-3 Anchor function, the SRv6 node allocates one UL L3-anchor segment SID per L3-anchor function. Each L3-anchor SID represents one policy which is shared by multiple sessions, such as a routing table, or a service policy with in the table. The routing table should maintain forwarding entries of the belonging MNs.

Applicable SRv6 functions and behaviors are determined by use cases described in Section 6.

5.4. Stateless Interworking

Stateless interworking function provides SRv6 node a role to interworking between existing mobile user-plane and SRv6 mobile user-plane. Figure 3 shows the SRv6 SID format for stateless interworking function that is encoding identifiers of corresponding tunnel in existing network as argument of the SID.

```
+----------------------+-------+-------+-------+
|    IW-IPv6-Prefix    |IPv4DA |IPv4SA |TUN-ID |
+----------------------+-------+-------+-------+
        128-a-b-c          a      b       c
```

Figure 3: Stateless Interworking SID Encoding

Stateless interworking function introduce following SRv6 end function and transit behavior.

End.TM:	End point function with encapsulation for mapped tunnel
T.Tmap:	Transit behavior with tunnel decapsulation and mapping an SRv6 Policy

Stateless interworking function is associated with the following mandatory parameters:

IW-IPv4-Prefix:	IPv4 prefix representing network of SRv6 user-plane for legacy mobile user-plane
IW-IPv6-Prefix:	IPv6 prefix representing network of legacy mobile user-plane for SRv6 user-plane
TUN-PROTO:	Tunnel protocol type, such as GTP-U or GRE for PMIP

5.4.1. End.TM: End point function with encapsulation for mapped tunnel

The "End point to encapsulate for mapped tunnel" function (End.TM for short) is used to the direction from SRv6 user-plane to legacy user-plane network.

When interworking node N receives a packet destined to S and S is a local End.TM SID, N does:

```
 1. IF NH=SRH & SL > 0 THEN
 2.    decrement SL
 3.    update the IPv6 DA with SRH[SL]
 4.    push header of TUN-PROTO with tunnel ID from S            ;; Ref1
 5.    push outer IPv4 header with SA, DA from S
 6. ELSE
 7.    Drop the packet
```

Ref1: TUN-PROTO indicates target tunnel type.

5.4.2. T.Tmap: Transit behavior with tunnel decapsulation and mapping an SRv6 Policy

The "Transit with tunnel decapsulation and map to an SRv6 policy" function (T.Tmap for short) is used to the direction from legacy user-plane to SRv6 user-plane network.

When interworking node N receives a packet destined to a IW-IPv4-Prefix, N does:

```
 1.   IF P.PLOAD == TUN-PROTO & T.PLOAD == IPv6 THEN    ;; Ref1, Ref1bis
 2.      pop the outer IPv4 header and tunnel headers
 3.      copy IPv4 DA, SA, TUN-ID to form SID B with IW-IPv6-Prefix
 4.      insert the SRH (D, B; SL=1)                    ;; Ref2, Ref2bis
 5.      set the IPv6 DA = B
 6.      forward along the shortest path to B
 7.   ELSE
 8.      Drop the packet
```

Ref1: P.PLOAD and T.PLOAD represent payload protocol of the receiving packet, and payload protocol of the tunnel respectively.

Ref1bis: First nibble of payload is used to determine payload protocol in GTP-U case due to it has no payload protocol indicator in the header.

Ref2: The received IPv6 DA is placed as last SID of the inserted SRH.

Ref2bis: The SRH is inserted before any other IPv6 Routing Extension Header.

5.5. Rate Limit

Mobile user-plane requires rate-limit feature. SID is able to encode limiting rate as an argument in SID. Multiple flows of packets should have same group identifier in SID when those flows are in an same AMBR group. This helps to keep user-plane stateless. That enables SRv6 endpoint nodes which are unaware from the mobile control-plane information. Encoding format of rate limit segment SID is following:

```
+----------------------+----------+-----------+
| Locater of rate-limit| group-id | limit-rate|
+----------------------+----------+-----------+
          128-i-j           i          j
```

Figure 4: Stateless Interworking SID Encoding

In case of j bit length is zero in SID, the node should not do rate limiting unless static configuration or control-plane sets the limit rate associated to the SID.

6. Segment Routing IPv6 Functions and Behaviors by Use Cases

This section describes SRv6 functions and behavior applied to mobile user-plane functions by use cases. Terminology of SRv6 endpoint functions refers to [I-D.filsfils-spring-srv6-network-programming].

6.1. Basic Mode

In the basic mode, we assume that mobile user-plane functions are as same as existing ones except using SRv6. This means that there is no impact to rest part of mobile system should be expected while no advanced segment routing features are introduced to it.

User-plane Function	Uplink	Downlink
Access Point	T.Insert	End.X
L2-anchor	End.B6	End.B6
L3-anchor	End.T	T.Insert

Table 1: SRv6 Functions for Basic Mode

6.1.1. Uplink

In uplink, SRv6 node applies following SRv6 end point functions and transit behavior. SIDs are allocated per L3-anchor function in the SRv6 nodes of both L2 and L3-anchor functions in basic mode.

Access Point:

When the access point node receives a packet destine to "D::1" from a mobile node "S::1", it does T.Insert process for the receiving packets to push a SRH with SID list <A2::1, D::1> with SL=1. The access point node update DA to next UL L2-anchor SID "A2::1" which the SL indicates and forward the packet.

Layer-2 Anchor:

The L2-anchor node of "A2::1" segment does End.B6 process for the receiving packet according to the SRH. The node updates DA to next UL L3-anchor SID "A3::1" bound to "A2::1" and forward the packet. In this basic use case, just one UL L3-anchor SID with SL=0 is enough to do it so that there is no need to push another SRH to the packet in that PSP (Penultimate Segment Pop) operation.

Layer-3 Anchor:

The L3-anchor node of "A3::1" segment does End.T process for the receiving packet according to the SRH. The node decrement SL to 0, updates DA to D::1 which the SL indicates and lookup IPv6 table associated with "A3::1". In this basic use case, the decremented SL is 0 so that the node does PSP operation of popped out the SRH from the packet and forward it.

6.1.2. Downlink

In downlink, SRv6 node applies following SRv6 end point functions and transit behavior. SIDs are allocated per session in the SRv6 nodes of both L2-anchor and access point functions in basic mode.

Layer-3 Anchor:

When the L3-anchor node receives a packet destine to "S::1" from a correspondent node "D::1", it does T.Insert process for the receiving packets to push a SRH with SID list <A2::2, S::1> with SL=1. The L3-anchor node update DA to next DL L2-anchor SID "A2::2" which the SL indicates and forward the packet.

Layer-2 Anchor:

The L2-anchor node of "A2::2" segment does End.B6 process for the receiving packet according to the SRH. The node updates DA to next DL access point segment "A1::1" bound to "A2::2" and forward the packet. In this basic use case, just one DL access point SID with SL=0 is enough to do it so that there is no need to push another SRH to the packet in that PSP (Penultimate Segment Pop) operation.

Access Point:

The access point node of "A1::1" segment does End.X process for the receiving packet according to the segment. The node decrement SL to 0, updates DA to S::1 which the SL indicates and forward the packet to the mobile node of "S::1" through radio channel associated with "A1::1". In this use case, the decremented SL is 0 so that the node does PSP operation of popped out the SRH from the packet and forward it.

6.2. Aggregate Mode

In the aggregate mode, user-plane function is able to steer multiple mobile sessions per service policy. This means that mobile sessions paths are aggregated into a service path which includes not only mobile user-plane functions but also other nodes, links or service functions. SIDs are allocated per service policy in the SRv6 nodes of user-plane functions in the aggregate mode.

Aggregate mode user-plane can take advantage of SRv6 that enables seamless mobile user-plane deployment with service chaining, VPNs, traffic-engineering by computed path to fulfil the policy.

User-plane Function	Uplink	Downlink
Access Point	T.Insert	End
L2-anchor	End or End.B6	End or End.B6
L3-anchor	End.T	T.Insert

Table 2: SRv6 Functions for Aggregate Mode

6.2.1. Uplink

In uplink, SRv6 node applies following SRv6 end point functions and transit behavior.

Access Point:

When the access point node receives a packet destine to "D::1" from a mobile node "S::1", it does T.Insert process for the receiving packets.

First scenario is that the service policy for "D::1" is via service function S1 and node C1 before reach to L2-anchor "A2::1" so that the node pushes a SRH with SID list <S1, C1, A2::1, D::1> with SL=3. The node updates DA to service function S1 which the SL indicates and forward the packet.

Second case is that the access point function directly indicates the path beyond the L2-anchor, service function S2 and L3-anchor "A3::1" for example, the SID list shoud be <S1, C1, A2::, S2, A3::1, D::1> with SL=5.

Layer-2 Anchor:

It is assumed that the packet successfully traversed S1 and C1 segments so that the SL was decremented 3 to 1 in the first scenario, and 5 to 3 in the second scenario before arriving the node of "A2::1" or "A2::" segment.

In the first scenario that the L2-anchor node of "A2::1" segment is bound to a service policy which indicates path via service function S2 and L3-anchor function A3::1, the node does End.B6 process for the receiving packet to push a new SRH with SID list <S2, A3::1> with SL=1. The node updates DA to next service function SID S2 which the SL indicates and forward the packet.

In the second scenario that one SRH with SIDs <S1, C1, A2::, S2, A3::1, D::1>, the L2-anchor node of "A2::" segment is bound to just End function. The node does End process for the receiving packet according to the SRH that decrements SL to 2, updates DA to next service function SID S2 which the SL indicates and forward the packet.

Layer-3 Anchor:

The L3-anchor node of "A3::1" segment does End.T process for the receiving packet according to the SRH(s).

In the first scenario that outer SRH with SIDs <S2, A3::1>, it is assumed that the SL is decremented to 0 at service function S2 so that the node pops outer SRH. Then the node processes second SRH with SIDs <S1, C1, A2::1, D::1> that decrements SL to 0, updates DA to D::1 which the SL indicates and lookup IPv6 table associated with "A3::1". In this case, the decremented SL is 0 so that the node does PSP operation of popped out the SRH from the packet and forward it.

In the second scenario that one SRH with SIDs <S1, C1, A2::, S2, A3::1, D::1>, L3-anchor node of "A3::" segment does End.T process for the receiving packet according to the SRH. Rest part of processes are as same as previous case.

6.2.2. Downlink

In downlink, SRv6 node applies following SRv6 end point functions and transit behavior.

Layer-3 Anchor:

When the L3-anchor node receives a packet destine to "S::1" from a mobile node "D::1", it does T.Insert process for the receiving packets.

First scenario is that the service policy for "S::1" is via service function S2 before reach to L2-anchor "A2::2" so that the node pushes a SRH with SID list <S2, A2::2, S::1> with SL=2. The node updates DA to service function S2 which the SL indicates and forward the packet.

Second scenario is that the L3-anchor function directly indicates the path beyond the L2-anchor, service function S1, node C3 and access point "A1::" for example, the SID list shoud be <S2, A2::, S1, C1, A1::1, D::1> with SL=5.

Layer-2 Anchor:

It is assumed that the packet successfully traversed S2 segment so that the SL was decremented 2 to 1 in the former case, and 5 to 4 in the latter case before arriving the node of "A2::2" or "A2::" segment.

In the first scenario that the L2-anchor node of "A2::2" segment is bound to a service policy which indicates path via node C1, service function S1 and access point function A1::, the node does End.B6 process for the receiving packet to push a new SRH with SID list <C1, S1, A1::1> with SL=1. The node updates DA to next service function SID C1 which the SL indicates and forward the packet.

In the second scenario that one SRH with SIDs <S1, C1, A2::, S2, A3::1, D::1>, the L2-anchor node of "A2::" segment is bound to just End function. The node does End process for the receiving packet according to the SRH that decrements SL to 2, updates DA to next node C1 which the SL indicates and forward the packet.

Access Point:

The access point node of "A1::1" segment does End process for the receiving packet according to the SRH(s).

In the first scenario that outer SRH with SIDs <C1, S1, A1::1>, it is assumed that the SL is decremented 2 to 0 at node C1 and service function S1 so that the outer SRH is popped out by PSP at S1. Thus the node processes second SRH with SIDs <S2, A2::2, S::1> that decrements SL to 0, updates DA to S::1 which the SL indicates and forward the packet to the mobile node through radio channel associated with "S::1". In this case, the decremented SL is 0 so that the node does PSP operation of popped out the SRH from the packet and forward it.

In the second scenario that one SRH with SIDs <S2, A2::, C1, S1, A1::1, S::1>, access node of "A1::1" segment does End process for the receiving packet according to the SRH. Rest part of processes are as same as previous case.

6.3. Stateless Interworking with Legacy Access Network

This section describes an use case where user-plane functions are interworking in stateless between SRv6 and legacy access networks. Here stateless means that there’s no need to be aware any states of mobility sessions in the node.

As some types of interworking scenarios could be considered, we will describe other cases in the future versions of this document.

6.3.1. Uplink: Lagacy Access to SRv6

Legacy Access Point:

When a legacy access point node receives a packet destined to "D::1" from a mobile node "S::1" through associated radio channel, it does tunnel encapsulation with the tunneling parameters, IPv4 DA, SA and tunnel header with ID, and sends out the packet to the network.

Stateless Interworking:

The stateless interworking node of corresponding to the IPv4 DA does T.Tmap process for the receiving packet to pop the tunnel headers and push a SRH to the packet. The SRH consists of SIDs <B1, D::1> with SL=1, where "B1" encodes IW-IPv6-Prefix, tunnel IPv4 DA, SA and TUN-ID in it as Figure 3 defined. The stateless interworking node updates DA to "B1" and forward the packet. SA of the packet must be kept as "S::1".

Layer-2 or Layer-3 Anchor:

The receiving node of the packet destine to B1 either be Layer-2 anchor, or Layer-3 anchor node. Which type of anchor function bound to B1 depends on operational policy.

In case of B1 to an L2-anchor node, the L2-anchor node does End.B6 process for the receiving packet as same as previous section.

In case of B1 to an L3-anchor node, the L3-anchor node does End.T process for the receiving packet as same as previous section.

6.3.2. Downlink: SRv6 to Legacy Access

Layer-2 or Layer-3 Anchor:

In case of an L3-anchor node receives a packet destine to S::1 from the correspondent node "D::1", and stateless interworking SID is bound to the next segment of S::1 , the L3-anchor node does T.Insert process for the receiving packet to push a SRH with SID list <B2::, S::1> with SL=1, where "B2" which encodes IW-IPv6-Prefix, tunnel IPv4 DA, SA and TUN-ID in it as Figure 3 defined. The L3-anchor node updates DA to "B2" and forward the packet.

In case of an L2-anchor node receives a packet destine to SID "A2::B2" and the SID is bound to "B2", the L2-anchor node does End.B6 process for the receiving packet as same as previous section. The node updates DA to B2 and forward the packet.

Stateless Interworking:

The stateless interworking node of "B2" End.TM process for the receiving packet according to the SRH. The node decrement SL to 0, updates DA to "D::1" which the SL indicates, push IPv4 and tunnel headers with IPv4 DA, SA and TUN-ID extracted from "B2", and forward the packet to the legacy network.

In this use case, decremented SL is 0 so that the node does PSP operation of popped out the SRH from the packet and forward it.

Legacy Access Point:

The legacy access point node of the IPv4 DA does tunnel termination process for the receiving packet according to the tunnel header. The node pop the IPv4 and tunnel header and forward the packet to the mobile node through radio channel associated with the tunnel.

7. Network Slicing Considerations

Mobile network may be required to create a network slicing that represent a set of network resources and isolate those resource from other slices. User-plane functions represented as SRv6 segments would be part of a slice.

To represent a set of user-plane function segments for a slice, sharing same prefix through those SIDs within the slice could be a straightforward way. SIDs in a network slice may include other type of functions in addition to the mobile user-plane functions described in this document, and underlay integration to meet SLA and quality requirements.

While network slicing has been discussed in the IETF and other standardization bodies, what functionalities are required for network slicing in mobile user-plane is further study item and to be discussed.

8. Control Plane Considerations

8.1. Existing Control Plane

A mobile control-plane entity may allocate SIDs to the node of corresponding user-plane function. In this case, the control-plane entity must signal allocated SIDs to other side of entity.

If the control-plane entity needs to employ existing mobile control-plane protocol to signal, GTP-C or PMIP for example, it must require not to impact the control-plane protocols. In this case, tunnel endpoint IPv6 address field in the control-plane message can be used to signal a SID.

The basic mode described in Section 6.1 should be adopted. In the basic mode, SID indicates unique session so that tunnel identifier is not required to SRv6 user-plane. But the mobile control-plane may be assumed that one user-plane entity has one IPv6 address and it allocates tunnel identifier per session. In this case, SRv6 node of user-plane can form SID with the IPv6 address and the tunnel identifier.

When an IPv6 prefix "A::/64" is allocated to an user-plane, and the control-plane allocate one 32bits tunnel identifier of "0x12345678" for a mobile session, the user-plane node forms a 128bits SID "A::1234:5678".

In this way, there is no impact to the control-plane.

8.2. Aggregate Mode

To support aggregate mode described in Section 6.2 in control-plane protocol, allocated SIDs for service policies can be signaled as tunnel endpoint IPv6 address too. In this case, SRv6 policy associated to the SID should be configured in the user-plane node through other means.

Aggregate mode user-plane can take advantage of SRv6 that enables seamless mobile user-plane deployment with service chaining, VPNs, traffic-engineering by computed path to fulfil the policy.

In case of a mobile control-plane is aware those policies and is capable to advertise them, the mobile control-plane can integrate SRv6 advanced features.

8.3. User-Plane Sepalated Control Plane

In an user-plane separated control-plane system, a mobile user-plane entity may allocate SIDs to an user-plane function instead of the control-plane. In this case, the user-plane entity should inform allocated SIDs back to the control-plane entity.

If the control- and user-plane entities need to employ existing user-plane control protocol in between, such as PFCP for example, it may be required not to impact that protocol. In this case, the control-plane entity is only allowed to signal SID in a tunnel endpoint IPv6 address field.

8.4. Centralized Controller

When a centralized controller interfaces to mobile control-planes is capable to allocate SIDs to the controlling SRv6 nodes, the mobile control-planes just need to indicate nodes and their user-plane functions to the controller. In this case, the controller must allocate appropriate SIDs for the user-plane functions to the SRv6 nodes. The controller must configure allocated SIDs to the nodes.

To indicate nodes and their user-plane functions from mobile control-plane to user-plane, the centralized controller could take advantage of [I-D.ietf-dmm-fpc-cpdp]. It provides interface to the control-plane to manage the user-plane of mobile networks. To build centralized controller for mobile user-plane is out of scope of this document.

8.5. Stateless Interworking

A mobile control-plane is not required to configure statless interworking function in interworking node. This benefits operators to gradually migrate from legacy to advanced mobile user-plane with no impact to the legacy system.

SID allocating entity of SRv6 user-plane nodes needs to be aware of IW-IPv6-Prefix to form interworking SID. Legacy user-plane network allocate IPv4 address space routed to SRv6 user-plane network. The mobile control-plane of legacy user-plane also need to allocate tunnel endpoint IPv4 addresses from that address space for mobile sessions which is usual operation for it and no difference from legacy system.

9. Security Considerations

TBD

10. IANA Considerations

This document has no actions for IANA.

11. Acknowledgements

TBD

12. References

12.1. Normative References

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Matsushima, S., Lebrun, D., Decraene, B., Peirens, B., Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P., Sharif, M., Ayyangar, A., Mynam, S., Henderickx, W., Bashandy, A., Raza, K., Dukes, D., Clad, F., and P. Camarillo, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-02 (work in progress), October 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-13 (work in progress), October 2017.

[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>.

12.2. Informative References

[I-D.ietf-dmm-fpc-cpdp]
Matsushima, S., Bertz, L., Liebsch, M., Gundavelli, S., Moses, D., and C. Perkins, "Protocol for Forwarding Policy Configuration (FPC) in DMM", draft-ietf-dmm-fpc-cpdp-09 (work in progress), October 2017.

[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>.

[TS.29281]
3GPP, , "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", 3GPP TS 29.281 10.3.0, September 2011.

Authors’ Addresses

Satoru Matsushima
SoftBank
Tokyo
Japan

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

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Miya Kohno
Cisco Systems, Inc.
Japan

Email: mkohno@cisco.com

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 10, 2018

D. Moses
W. Feng
Intel
A. Yegin

August 9, 2017

DHCPv6 Extension for On Demand Mobility exposure
draft-moses-dmm-dhcp-ondemand-mobility-08

Abstract

Applications differ with respect to whether or not they need IP session continuity and/or IP address reachability. Networks providing the same type of service to any mobile host and any application running on the host yields inefficiencies. This document describes extensions to the DHCPv6 protocol to enable mobile hosts to indicate the required mobility service type associated with a requested IP prefix and to allow networks to indicate the type of mobility service associated with the allocated IP prefix in return.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-dmm-ondemand-mobility] defines different types of mobility-associated services provided by access networks to mobile hosts with regards to maintaining IPv6 prefix continuity after an event of the host moving between locations with different points of attachments within the IP network topology. It further specifies means for applications to convey to the IP stack in the mobile host, their requirements regarding these services.

This document defines extensions to the DHCPv6 protocol ([RFC3315]) and [RFC3633] in the form of a new DHCP option that specifies the type of mobility services associated with an IPv6 prefix. The IP stack in a mobile host uses the DHCP client to communicate the type of mobility service it wishes to receive from the network. The DHCP server in the network uses this option to convey the type of service that is guaranteed with the assigned IPv6 prefix in return.

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 , [RFC2119] [RFC8174] when, they appear in all capitals, as shown here.

3. IPv6 Continuity Service Option

The IPv6 Continuity Service option is used to specify the type of continuity service associated with a source IPv6 prefix. The IPv6 Continuity Service option MUST be encapsulated in the IAprefix-options field of the IA_PD prefix option.

The format of the IPv6 Continuity Service option is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OPTION_IPv6_CONTINUITY_SERVICE|         option-length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| service-type  |
+-+-+-+-+-+-+-+-+
```

option-code	OPTION_IPv6_CONTINUITY_SERVICE (TBD)
option-len	1
service-type	one of the following values:

Non-Persistent -	a non-persistent IP prefix (1)
Session-Lasting -	a session-lasting IP prefix (2)
Fixed -	a fixed IP prefix (3)
Graceful-replacement -	a graceful-replacement IP prefix (4)
Anytype -	Anyone of the above (0)

The definition of these service types is available in [I-D.ietf-dmm-ondemand-mobility].

All other values (5-255) are reserved for future use. If the OPTION_IPv6_CONTINUITY_SERVICE option is received and its service-type is equal to one of the reserved values, the option SHOULD be ignored.

When a message is sent from a client to a server, the value of the IPv6 Continuity Service option indicates the type of continuity service required for the IPv6 prefix requested by the client.

When a message is sent from a server to a client, the value of the IPv6 Continuity Service option indicates the type of continuity service committed by the network for the associated IPv6 prefix. The value 'AnyType' SHOULD only appear in the message sent from the client to the server to indicate that the client has no specific preference. However, it cannot appear in a message sent from the server.

Once an IPv6 prefix type is requested and provided, any subsequent messages involving this prefix (lease renewal - for example) MUST include the IPv6 Continuity Service option with the same service type that was assigned by the server during the initial allocation.

If a server receives a request to assign an IPv6 prefix with a specified IPv6 Continuity service, but cannot fulfill the request, it MUST reply with the NoPrefixAvail status.

A server that does not support this option will ignore it and respond without taking into account the desired session continuity service. The response will not include the Continuity Service option encapsulated in the IAprefix-options field of the IA_PD prefix option.

The missing Continuity Service option in the response serves as an indication to the client that this feature is not supported by the server. It MAY use the allocated prefix knowing it does not necessarily support the desired Continuity service, or perform any other action.

A server MUST NOT include the IPv6 Continuity Service option in the IAprefix-options field of an IA_PD Prefix option, if not specifically requested previously by the client to which it is sending a message.

If a client receives an IA_PD Prefix option from a server with the IPv6 Continuity Service option in the IAprefix-options field, without initially requesting a specific service using this option, it MUST discard the received IPv6 prefix.

If the mobile device (host or router) has no preference regarding the type of continuity service it uses the 'AnyType' value as the specified type of continuity service. The Server will allocate an IPv6 prefix with some continuity service and MUST specify the type in IPv6 Continuity Service option encapsulated in the IAprefix-options field of the IA_PD Prefix option. The method for selecting the type of continuity service is outside the scope of this specification.

4. Correlation between Session Continuity Service and Lifetime Values

The values to be used in the Preferred-lifetime and Valid-lifetime fields in the IA Prefix Option are out of the scope of this specification and left to implementation. It is RECOMMENDED to provide longer lifetime values for Fixed and Session-lasting prefixes compared to the lifetime values of Non-persistent and Graceful-replacement prefixes because the network has guaranteed their validity regardless of the link to which the host is attached.

For clients using Graceful-replacement services, the network MAY obsolete a Prefix and allocate a new one from time to time especially in a mobility-related event. On such occasions, the network SHOULD provide a graceful period (lifetime) in which the obsoleted prefix can still be used and a new (longer) lifetime with the new prefix.

It is NOT RECOMMENDED using 0xFFFFFFFFFF (infinity) values for the lifetime of Fixed prefixes. Even though they are fixed, it is still safer to Rebind periodically. The lifetime value can be relatively long to reduce message exchange overhead.

Section 18.2 - Client Behavior of [I-D.ietf-dhc-rfc3315bis] specifies that when a client detects that it may have moved to a new link, it uses Rebind if it has delegated prefixes. It is worth clarifying that a client does not HAVE to Rebind the prefixes if they are Fixed or Session-lasting prefixes.

5. Security Considerations

There are no specific security considerations for this option.

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, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <http://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[I-D.ietf-dhc-rfc3315bis]
Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) bis", draft-ietf-dhc-rfc3315bis-09 (work in progress), June 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-06 (work in progress), July 2017.

[I-D.ietf-dmm-ondemand-mobility]
Yegin, A., Moses, D., Kweon, K., Lee, J., Park, J., and S. Jeon, "On Demand Mobility Management", draft-ietf-dmm-ondemand-mobility-12 (work in progress), July 2017.

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

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

[RFC7934] Colitti, L., Cerf, V., Cheshire, S., and D. Schinazi, "Host Address Availability Recommendations", BCP 204, RFC 7934, DOI 10.17487/RFC7934, July 2016, <http://www.rfc-editor.org/info/rfc7934>.

Authors’ Addresses

Danny Moses
Intel
Petah Tikva
Israel

Email: danny.moses@intel.com

Wu-chiX Feng
Intel
Hillsboro
USA

Email: wu-chix.feng@intel.com

Alper Yegin
Istanbul
Turkey

Email: alper.yegin@yegin.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018
October 30, 2017

B. Sarikaya
L. Xue

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 Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 2, 2018

Z. Yan
G. Geng
CNNIC
J. Lee
Sangmyung University
H. Chan
Huawei Technologies

October 29, 2017

Mobility Capability Negotiation and Protocol Selection
draft-yan-dmm-man-02

Abstract

The draft analyzes the issue that multiple mobility management protocols have been developed according to different requirements. These different protocols have different functional requirements on the network element or the host. A scheme is then proposed to support the negotiation and selection of adopted mobility management protocol when a host accesses a new network.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

A large number of multiple protocols have been developed. In order to clearly analyze the possible cases, these mobility management protocols can be categorized as follows:

o Mobile IPv6 (MIPv6) protocol: the mobility management scheme based on [RFC6275].
o Proxy Mobile IPv6 (PMIPv6) protocol: the mobility management scheme based on [RFC5213].
o MIPv6 suite protocols: based on MIPv6, there are multiple extension protocols have been standardized. These protocols can be classified into two types: protocols for functional extension and protocols for performance enhancement. The protocols for functional extension are proposed to support some specific scenarios or functions, such as Dual-stack Mobile IPv6 (DSMIPv6) [RFC5555] for mobility of the dual-stack nodes, Multiple Care-of-address (MCoA) [RFC5648] for hosts with multiple access interfaces and Network Mobility (NEMO) [RFC3963] for mobility of sub-network. The other type is proposed to enhance performance of the mobility management, such as Fast Mobile IPv6 (FMIP6) [RFC5268] for fast handover, Hierarchical Mobile IPv6 (HMIPv6) [RFC5380] for hierarchical mobility optimization. In the MIPv6 suite protocols, location update is initiated by the host and the tunnel is also terminated at the host.
o PMIPv6 suite protocols: in order to reduce the protocol cost and enhance the handover performance further, the network-based mobility management protocols were proposed and PMIPv6 was

      standardized as a base protocol. Based on PMIPv6, a series of its
      extensions were proposed, such as Dual-stack Proxy Mobile IPv6
      (DS-PMIPv6) [RFC5844], and Distributed Mobility Management Proxy
      Mobile IPv6 (DMM-PMIPv6) [RFC7333]. Being different from the
      MIPv6 suite protocols, the location update in PMIPv6 suite
      protocols is triggered by the network entity and the tunnel is
      established between network entities. Then the host needs to do
      nothing about signaling exchange during the movement,
      particularly, the mobility support is transparent to the IP layer
      of the host.

   o Network-based protocols: generally, they refer to the mobility
      management protocols which do not require the involvement of the
      host to support mobility. They include the PMIPv6 suite protocols
      and other network-based solutions, such as GPRS Tunnelling
      Protocol (GTP) [TS.29274][TS.29281].

   o Host-based protocols: generally, they refer to the mobility
      management protocols which require the involvement of the host in
      order to support mobility. They include the MIPv6 suite protocols
      and other host-based solutions, such as Host Identity Protocol
      (HIP) [RFC7401] and IKEv2 Mobility and Multihoming Protocol
      (MOBIKE) [RFC4555].

Figure 1 illustrates the scopes of the above different categories.

```
    +-----------------+        +-----------------+
    | Network-based   |        | Host-based      |
    |+---------------+|        |+---------------+|
    ||PMIPv6 suite   ||        ||MIPv6 suite    ||
    ||+-------------+||        ||+-------------+||
    |||PMIPv6       |||        |||MIPv6        |||
    ||+-------------+||        ||+-------------+||
    |+---------------+|        |+---------------+|
    +-----------------+        +-----------------+
```

Figure 1: Scopes of different protocol categories

In deployment, the host-based protocols and network-based protocols will be co-existing and multiple protocol deamons will be configured on the network entities or host. There is then a gap in how to determine which protocol to use. A scheme is therefore needed to support the negotiation and selection of mobility management protocol when the host initially attaches or hands over to a new network [Paper-CombiningMobilityStandards].

This document tries to present the principles for the protocol selection and analyze the possible scenarios which should be supported by the subsequent mobility solution.

2. Motivations

As illustrated above, these protocols may co-exist in practice and may simultaneously be used in an access network or even the same entity. Due to their different requirements on the network entity or host, a scheme is needed to support the negotiation and selection of adopted mobility management protocol when the host accesses to a new network. Generally, two problems should be solved:

- o What principles should be followed for the protocol negotiation and selection?
- o What procedure should be adopted for the protocol negotiation and selection?

This scheme is needed because the network entity and the host may have different capabilities and preferences (may be decided by the capability and mobility pattern of the host). This scheme aims to guarantee that the optimum and most suitable protocol will be used.

3. Possible Cases

From both host and network aspects, there are multiple cases in their capacities of mobility management as shown in Figure 2. We mainly analyze the cases where that host and network support a single protocol. If multiple protocols are supported simultaneously by the host or network side, multiple cases exist at the same time but the logic is the same as that in the case with single protocol supported. Specifically, the following cases should be considered.

1) Network supports network-based protocol, host supports network-based protocol

In this case, there are the following sub-cases:

a) Host supports PMIPv6 suite protocol, Network supports PMIPv6 suite protocol

- o if host supports PMIPv6 and network supports PMIPv6, PMIPv6 will be selected.
- o if host supports PMIPv6 and network supports extended PMIPv6 protocol, extended PMIPv6 is selected if no host involvement is needed, otherwise the plain PMIPv6 is selected (we assume that the extension protocols are backward-compatible with the related plain protocol).
- o if host supports extended PMIPv6 protocol and network supports PMIPv6, PMIPv6 is selected (we assume that the extension protocols are backward-compatible with the related plain protocol).

   o if host supports extended PMIPv6 protocol and network supports
      extended PMIPv6 protocol, the identical extension protocol is
      selected, otherwise, the plain PMIPv6 is selected (we assume that
      the extension protocols are backward-compatible with the related
      plain protocol).

```
       +----------------+-------------+--------------------------------+
       |                |             |PMIPv6                          |
       |                |             |-------------------+------------+
       | Network-based  | PMIPv6 suite|                   | DS-PMIPv6  |
       |                |             |                   +------------+
       |                |             |PMIPv6 extensions  | FPMIPv6    |
       |                |             |                   +------------+
       |                |             |                   | DMM-PMIPv6 |
       |                |             |                   +------------+
       |                |             |                   | ...        |
       |                |-------------+-------------------+------------+
       |                | Others      |GTP                             |
       |                |             |--------------------------------+
       |                |             |...                             |
       +----------------+-------------+--------------------------------+
       |                |             |MIPv6                           |
       |                |             |-------------------+------------+
       | Host-based     | MIPv6 suite |                   | DS-MIPv6   |
       |                |             |                   +------------+
       |                |             |                   | FMIPv6     |
       |                |             |                   +------------+
       |                |             |MIPv6 extensions   | HMIPv6     |
       |                |             |                   +------------+
       |                |             |                   | NEMO       |
       |                |             |                   +------------+
       |                |             |                   | DMM-MIPv6  |
       |                |             |                   +------------+
       |                |             |                   | ...        |
       |                |-------------+-------------------+------------+
       |                | Others      |HIP                             |
       |                |             |--------------------------------+
       |                |             |MOBIKE                          |
       |                |             |--------------------------------+
       |                |             |...                             |
       +----------------+-------------+--------------------------------+
```

     Figure 2: Possible capacities of mobility support by the host and
                                  network

   b) Host supports PMIPv6 suite protocol, Network supports other
   network-based protocol

- o if host supports PMIPv6 and network supports other network-based protocol, other network-based protocol is selected if no host involvement is needed, otherwise failure.
- o if host supports extended PMIPv6 protocol and network supports other network-based protocol, other network-based protocol is selected if no host involvement is needed, otherwise failure.

c) Host supports other network-based protocol, Network supports PMIPv6 suite protocol

- o if host supports other network-based protocol and network supports PMIPv6, PMIPv6 is selected.
- o if host supports other network-based protocol and network supports extended PMIPv6 protocol, extended PMIPv6 protocol is selected if no host involvement is needed, otherwise failure.

d) Host supports other network-based protocol, Network supports other network-based protocol

- o the identical protocol is selected, otherwise follow network ability if the protocols are different.

2) Network supports network-based protocol, host supports host-based protocol

In this case, there are the following sub-cases:

a) Host supports PMIPv6 suite protocol, Network supports MIPv6 suite protocol

- o if host supports PMIPv6 and network supports MIPv6, failure.
- o if host supports PMIPv6 and network supports extended MIPv6 protocol, failure.
- o if host supports extended PMIPv6 protocol and network supports MIPv6, failure.
- o if host supports extended PMIPv6 protocol and network supports extended MIPv6 protocols, failure.

b) Host supports PMIPv6 suite protocol, Network supports other host-based protocol

- o if host supports PMIPv6 and network supports other host-based protocol, failure.
- o if host supports extended PMIPv6 protocol and network supports other host-based protocol, failure.

c) Host supports other network-based protocol, Network supports MIPv6 suite protocol

- o if host supports other network-based protocol and network supports MIPv6, failure.
- o if host supports other network-based protocol and network supports extended MIPv6 protocol, failure.

d) Host supports other network-based protocol, Network supports other host-based protocol

- o failure.

3) Network supports host-based protocol, host supports network-based protocol

In this case, there are the following sub-cases:

a) Host supports MIPv6 suite protocol, Network supports PMIPv6 suite protocol

- o if host supports MIPv6 and network supports PMIPv6, PMIPv6 is selected in default and MIPv6 is selected if host prefers it.
- o if host supports MIPv6 and network supports extended PMIPv6 protocol, extended PMIPv6 is selected in default, then PMIPv6 is selected with the lower priority and MIPv6 is selected if host prefers it.
- o if host supports extended MIPv6 protocol and network supports PMIPv6, PMIPv6 will be selected in default, then extended MIPv6 is selected if host prefers it and network also supports, otherwise MIPv6 is selected with the lowest priority.
- o if host supports extended MIPv6 protocol and network supports extended PMIPv6 protocol, extended PMIPv6 is selected in default, then PMIPv6 is selected, then extended MIPv6 is selected if host prefers and network also supports, otherwise MIPv6 is selected with the lowest priority.

b) Host supports MIPv6 suite protocol, Network supports other network-based protocol

- o if host supports MIPv6 and network supports other network-based protocol, other network-based protocol is selected if no host involvement is needed, otherwise failure.
- o if host supports extended MIPv6 protocol and network supports other network-based protocol, other network-based protocol is selected if no host involvement is needed, otherwise failure.

c) Host supports other host-based protocol, Network supports PMIPv6 suite protocol

- o if host supports other host-based protocol and network supports PMIPv6, PMIPv6 is selected in default, otherwise failure.
- o if host supports other host-based protocol and network supports extended PMIPv6 protocol, extended PMIPv6 protocol is selected if no host involvement is needed, otherwise failure.

d) Host supports other host-based protocol, Network supports other network-based protocol

- o other network-based protocol is selected if no host involvement is needed, otherwise failure.

4) Network supports host-based protocol, host supports host-based protocol

In this case, there are the following sub-cases:

a) Host supports MIPv6 suite protocol, Network supports MIPv6 suite protocol

- o if host supports MIPv6 and network supports MIPv6, MIPv6 is selected.
- o if host supports MIPv6 and network supports extended MIPv6 protocol, MIPv6 is selected.
- o if host supports extended MIPv6 protocol and network supports MIPv6, MIPv6 is selected.
- o if host supports extended MIPv6 protocol and network supports extended MIPv6 protocols, the identical protocol is selected, otherwise MIPv6 is selected.

b) Host supports MIPv6 suite protocol, Network supports other host-based protocol

- o if host supports MIPv6 and network supports other host-based protocol, failure.
- o if host supports extended MIPv6 protocol and network supports other host-based protocol, failure.

c) Host supports other host-based protocol, Network supports MIPv6 suite protocol

- o if host supports other host-based protocol and network supports MIPv6, failure.
- o if host supports other host-based protocol and network supports extended MIPv6 protocol, failure.

d) Host supports other host-based protocol, Network supports other host-based protocol

- o the identical other host-based protocol is selected, otherwise failure.

5) Network supports host-based protocol and network-based protocol, host supports host-based protocol and network-based protocol

- o follow the network based protocol in default if the host can support, otherwise select the protocol both network and host can support if host prefers.

4. Principles and Possible Procedure

Two different schemes may be used for the protocol negotiation and selection: host-initiated and network-initiated. Within the MIP/PMIP protocols, the priority of the function-extension protocols should be higher than the performance-enhancement protocols. Generally, the following principles should be followed:

- o Priority 1: Follow network ability
- o Priority 2: Follow host preference
- o Priority 3: Support the functional extensions
- o Priority 4: Support the performance enhancements
- o In default: network based scheme if it can be supported

And the general procedure for the protocol selection should be:

- o During initiation, network-based protocol may be used as a default mobility management protocol once the network supports it.
- o If the host prefers host-based protocols, a negotiation is executed to handover from network-based protocol to host-based protocol.
- o After initial attachment, a profile will be generated in the management store to record the selected or preferred protocol of this host.
- o When the handover happens, the network will check the selected or preferred protocol during the authentication process. But the network also needs to notify the host if the selected protocol cannot be supported herein.

5. Extensions

In order to fulfill the above principles, some extensions should be supported, for example:

1) Extended negotiation messages

The protocol negotiation may be included in the MN_ATTACH Function [MN-AR.IF] and the implementation may be based on a new signaling message or extended messages (e.g., ICMPv6, Diameter, and RADIUS). Besides these, some other protocols may also be used in some specified scenarios, such as extended IEEE 802.21 primitives.

2) Extended management store

When the host accesses the network, authentication should be executed before the mobility management service is provided. In order to support the mobility management protocol selection, a new information should be recorded by the network after the successful authentication during the initial attachment. The newly introduced information shows the selected mobility management protocol and should be updated when the used protocol changes.

6. Security Considerations

Generally, this function will not incur additional security issues. The detailed influence should be analyzed in the future.

7. IANA Considerations

A new ICMP option or authentication option or other signaling message may be used with a new code number.

8. References

8.1. Normative References

[MN-AR.IF]
Laganier, J., Narayanan, S., and P. McCann, "Interface between a Proxy MIPv6 Mobility Access Gateway and a Mobile Node", draft-ietf-netlmm-mn-ar-if-03, February 2008.

[RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, DOI 10.17487/RFC3963, January 2005, <https://www.rfc-editor.org/info/rfc3963>.

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

[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>.

[RFC5268] Koodli, R., Ed., "Mobile IPv6 Fast Handovers", RFC 5268, DOI 10.17487/RFC5268, June 2008, <https://www.rfc-editor.org/info/rfc5268>.

[RFC5380] Soliman, H., Castelluccia, C., ElMalki, K., and L. Bellier, "Hierarchical Mobile IPv6 (HMIPv6) Mobility Management", RFC 5380, DOI 10.17487/RFC5380, October 2008, <https://www.rfc-editor.org/info/rfc5380>.

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

[RFC5648] Wakikawa, R., Ed., Devarapalli, V., Tsirtsis, G., Ernst, T., and K. Nagami, "Multiple Care-of Addresses Registration", RFC 5648, DOI 10.17487/RFC5648, October 2009, <https://www.rfc-editor.org/info/rfc5648>.

[RFC5844] Wakikawa, R. and S. Gundavelli, "IPv4 Support for Proxy Mobile IPv6", RFC 5844, DOI 10.17487/RFC5844, May 2010, <https://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, <https://www.rfc-editor.org/info/rfc6275>.

[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>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <https://www.rfc-editor.org/info/rfc7401>.

[TS.29274]
"3GPP Evolved Packet System (EPS); Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C); Stage 3", 3GPP TS 29.274 8.10.0, June 2011.

[TS.29281]
"General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", 3GPP TS 29.281 10.3.0, September 2011.

8.2. Informative References

[Paper-CombiningMobilityStandards]
Oliva, A., Soto, I., Calderon, M., Bernardos, C., and M. Sanchez, "The costs and benefits of combining different IP mobility standards", Computer Standards and Interfaces, February 2013.

Authors' Addresses

Zhiwei Yan
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing 100190
China

Email: yan@cnnic.cn

Guanggang Geng
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing 100190
China

Email: ggg@cnnic.cn

Jong-Hyouk Lee
Sangmyung University
31, Sangmyeongdae-gil, Dongnam-gu
Cheonan
Republic of Korea

Email: jonghyouk@smu.ac.kr

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

Email: h.a.chan@ieee.org