DMM
Internet-Draft
Intended status: Informational
Expires: June 17, 2017

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

December 14, 2016

Distributed Mobility Anchoring
draft-ietf-dmm-distributed-mobility-anchoring-03

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 or network slice 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 prefix which is allocated from and is therefore 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.

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. Distributed mobility management solutions do not rely on a centrally deployed mobility anchor in the data plane [Paper-Distributed.Mobility]. As such, the traffic of a flow SHOULD be able to change from traversing one mobility anchor to traversing another mobility anchor as a mobile node (MN) moves, or when changing operation and management requirements call for mobility anchor switching, thus avoiding non-optimal 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]. 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 or network slices, different kinds of mobility support are possible depending on the needs. 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 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.

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 (NEMO) 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.

An MN attached to an access router of a network or network slice may be allocated an IP prefix which is anchored to that router. It may then use an IP address configured from this prefix as the source IP address to run a flow 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. Although the anchor is in the MN’s network of attachment when the flow was initiated, the MN may later move to another network, so that the IP 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 or network slice supporting a mix of flows requiring and not requiring IP mobility support will need to distinguish these flows. The guidelines for such network or network slice are described in Section 4.1.1. The general guidelines for such network or network slice 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. Mobility involving change in the Data Plane Node (DPN) without changing the Data Plane Anchor (DPA) is described in Section 5.3 with additional guidelines described in Section 5.3.1 Mobility involving change in the DPN without changing the DPA is described in Section 5.4 with additional guidelines described in Section 5.4.1

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 allocated the HoA used for 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.

IP prefix/address anchoring: An IP prefix, i.e., Home Network Prefix (HNP), or address, i.e., HoA, allocated to 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 allocated IP prefix.

Location Management (LM) function: managing and keeping track of the internetwork location of an MN. The location information may be a binding of the IP advertised 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 IP prefix

delegated to the MR. The MNP is allocated to the MNNs in the mobile network.

LM is a control plane function.

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).

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 to 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.

Security Management (SM) function: The security management function controls security mechanisms/protocols providing access control, integrity, authentication, authorization, confidentiality, etc. for the control plane and data plane.

This function resides in all nodes such as control plane anchor, data plane anchor, mobile node, and correspondent node.

3. Distributed Mobility Anchoring

3.1. Configurations for Different Networks or Network Slices

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 or different network slices 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 shows two different configurations of network-based mobility management for a flat network.

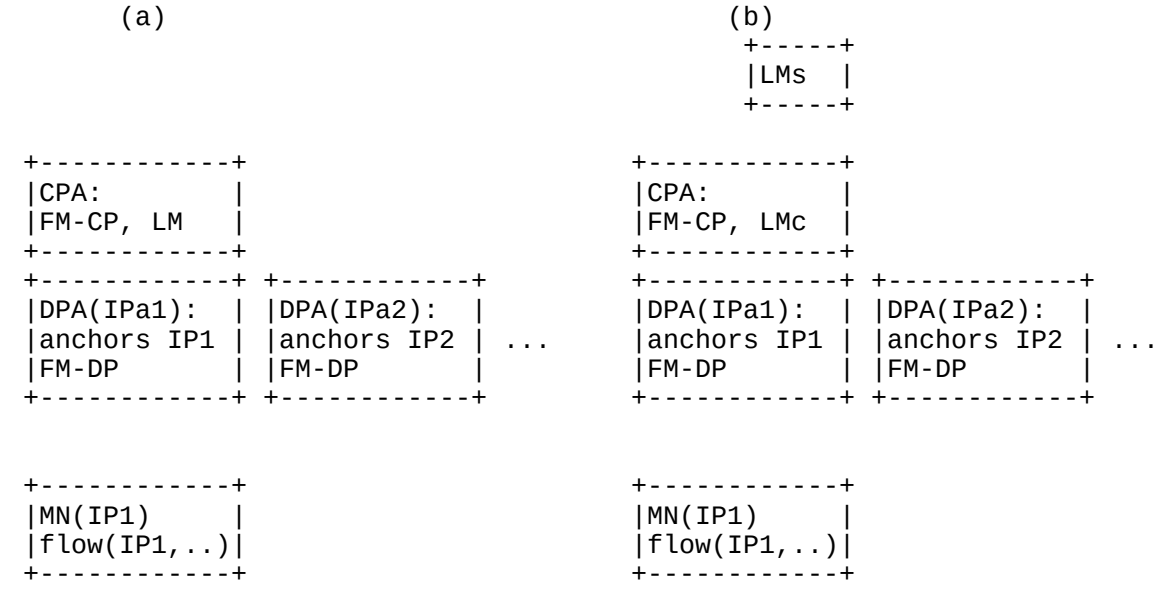


Figure 1. Configurations of network-based mobility management for a flat network (a) FM-CP and LM at CPA, FM-DP at DPA; (b) Separate LMs, FM-CP and LMc at CPA, FM-DP at DPA.

Figure 1 also shows a distributed mobility anchoring environment with multiple instances of the DPA.

There is an FM-DP function at each of the distributed DPA.

The control plane may either be distributed (not shown) or centralized. When the CPA co-locates with the distributed DPA there will be multiple instances of the co-located CPA and DPA (not shown).

There is an FM-CP function at the CPA.

An MN is allocated an IP prefix/address IP1 which is anchored to the DPA with the IP prefix/address IPa1. 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, ...) which uses IP1 and other parameters.

In Figure 1(a), LM and FM-CP co-locate at CPA.

Then LM may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

Figure 1(b) differs from Figure 1(a) in that the LM function is split into a server LMs and a client LMc.

LMc and FM-CP co-locate at the CPA.

The LMs may be centralized whereas the LMc may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

3.1.2. Network-based Mobility Support for a Hierarchical Network

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

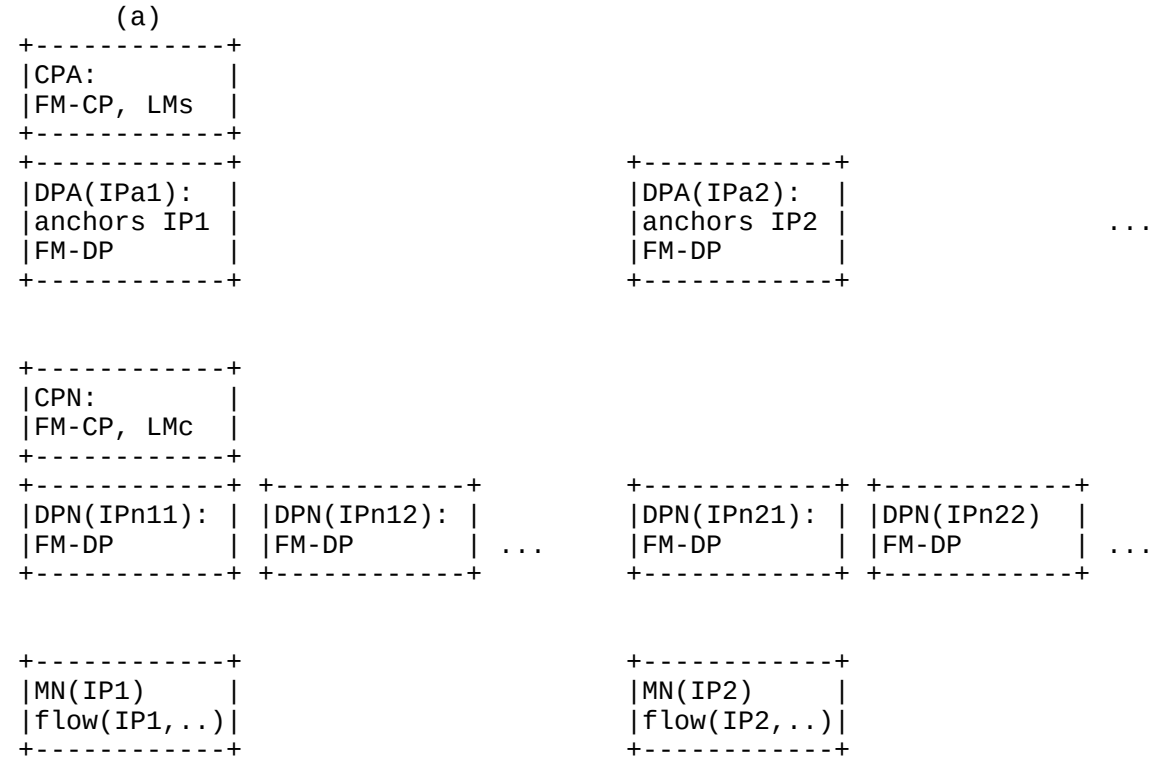


Figure 2(a). Configurations of network-based mobility management for a hierarchical network with FM-CP and LMs at CPA, FM-DP at DPA; FM-CP and LMc at CPN, FM-DP at DPN.

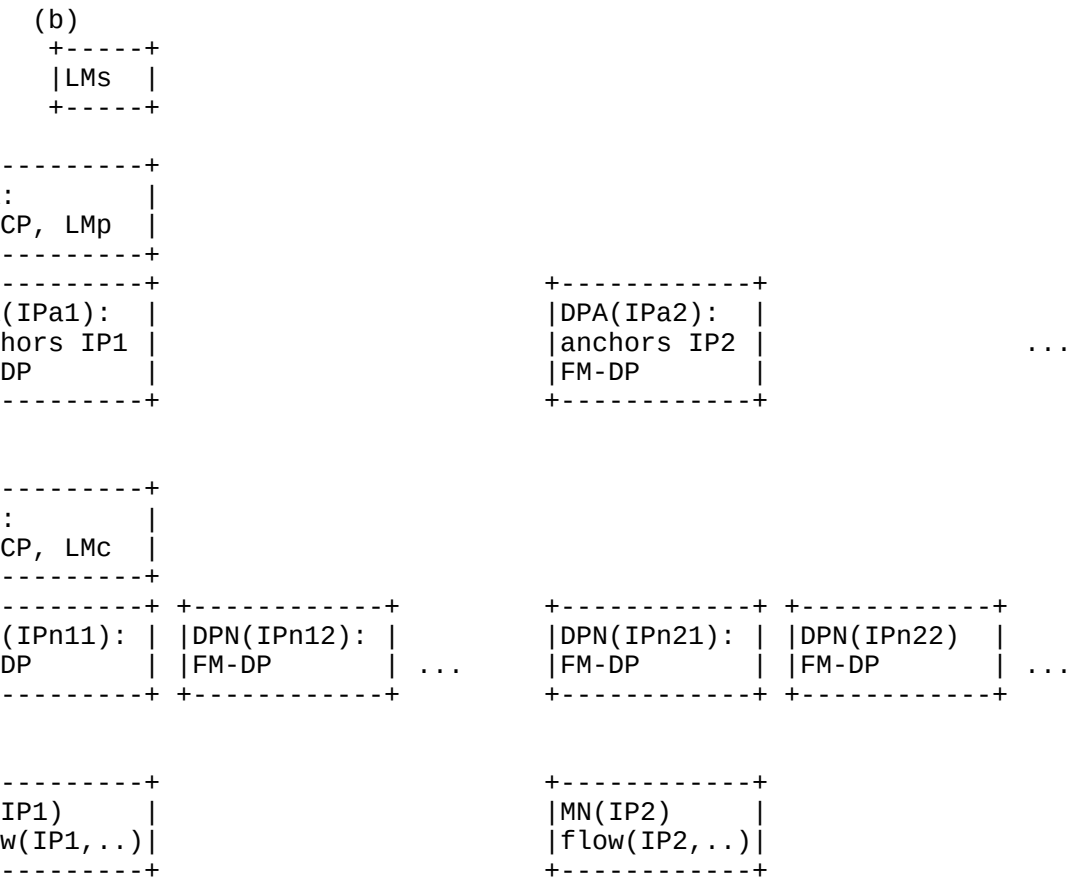


Figure 2(b). Configurations of network-based mobility management for a hierarchical network with separate LMs, FM-CP and LMp at CPA, FM-DP at DPA; FM-CP and LMc at CPN, FM-DP at DPN.

Figures 2 also shows a distributed mobility anchoring environment with multiple instances of the DPA.

In the hierarchy, there may be multiple DPN’s for each DPA.

There is FM-DP at each of the distributed DPA and at each of the distributed DPN.

The control plane may either be distributed (not shown) or centralized.

When the CPA co-locates with the distributed DPA there will be multiple instances of the co-located CPA and DPA (not shown).

When the CPN co-locates with the distributed DPN there will be multiple instances of the co-located CPN and DPN (not shown).

There is FM-CP function at the CPA and at the CPN.

MN is allocated an IP prefix/address IP1 which is anchored to the DPA with the IP prefix/address IPa1. It is using IP1 to communicate with a correspondent node (CN) not shown in the figure. The flow of this communication session is shown as flow(IP1, ...) which uses IP1 and other parameters.

In Figure 2(a), LMs and FM-CP are at the CPA. In addition, there are FM-CP and LMc at the CPN.

LMs may be distributed or centralized according to whether the CPA is distributed or centralized. The CPA may co-locate with DPA or may separate.

Figure 2(b) differs from Figure 2(a) in that the LMs is separated out, and a proxy LMp is added between the LMs and LMc.

LMp and FM-CP co-locate at the CPA.

FM-CP and LMc co-locate at the CPN.

The LMs may be centralized whereas the LMp may be distributed or centralized according to whether the CPA is distributed or centralized.

3.1.3. Host-based Mobility Support

Host-based variants of the mobility function configurations from Figures 2(a) and 2(b) are respectively shown in Figures 3(a) and 3(b) 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.

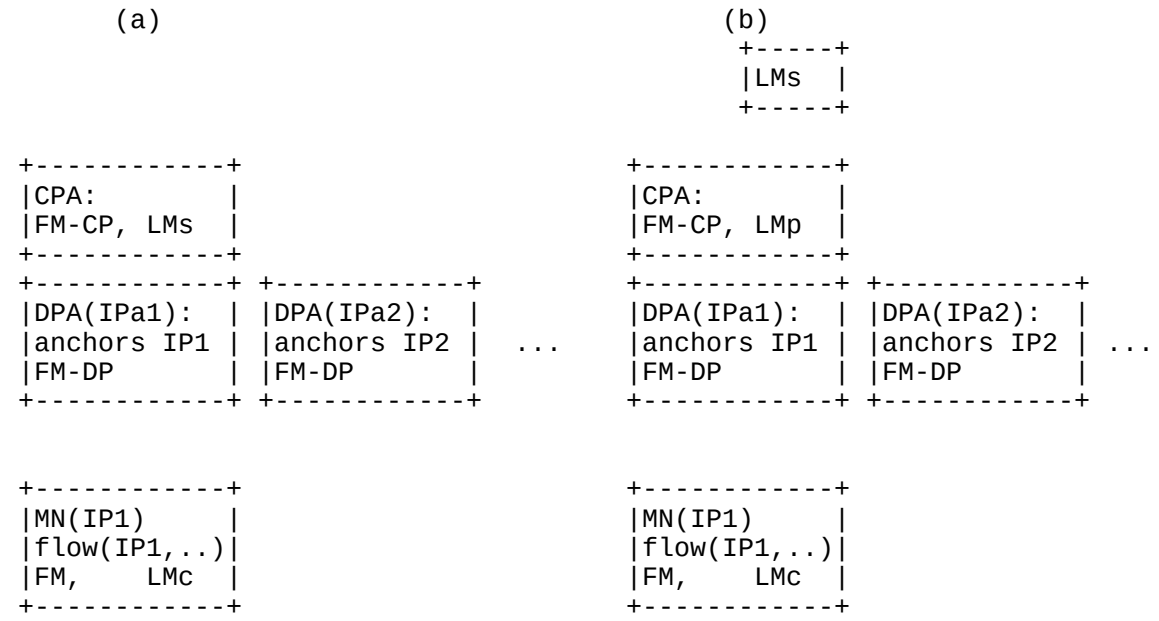


Figure 3. Configurations of host-based mobility management (a) FM-CP and LMs at CPA, FM-DP at DPA, FM and LMc at MN; (b) Separate LMs, FM-CP and LMp at CPA, FM-DP at DPA, FM and LMc at MN.

Figure 3 shows 2 configurations of host-based mobility management with multiple instances of DPA for a distributed mobility anchoring environment.

There is an FM-DP function at each of the distributed DPA.

The control plane may either be distributed (not shown) or centralized.

When the CPA co-locates with the distributed DPA there will be multiple instances of the co-located CPA and DPA (not shown).

There is an FM-CP function at the CPA.

The MN possesses the mobility functions such as FM and LMc.

The MN is allocated an IP prefix/address IP1 which is anchored to the DPA with the IP prefix/address IPa1. It is using IP1 to communicate with a CN not shown in the figure. The flow of this communication session is shown as flow(IP1, ...) which uses IP1 and other parameters.

In Figure 3(a), LMs and FM-CP co-locate at the CPA.

The LMs may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

Figure 3(b) differs from Figure 3(a) in that the LMs is separated out and the proxy LMp is added between the LMs and LMc.

LMp and FM-CP co-locate at the CPA.

The LMs may be centralized whereas the LMp may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

3.1.4. NEtwork MObility (NEMO) Basic Support

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

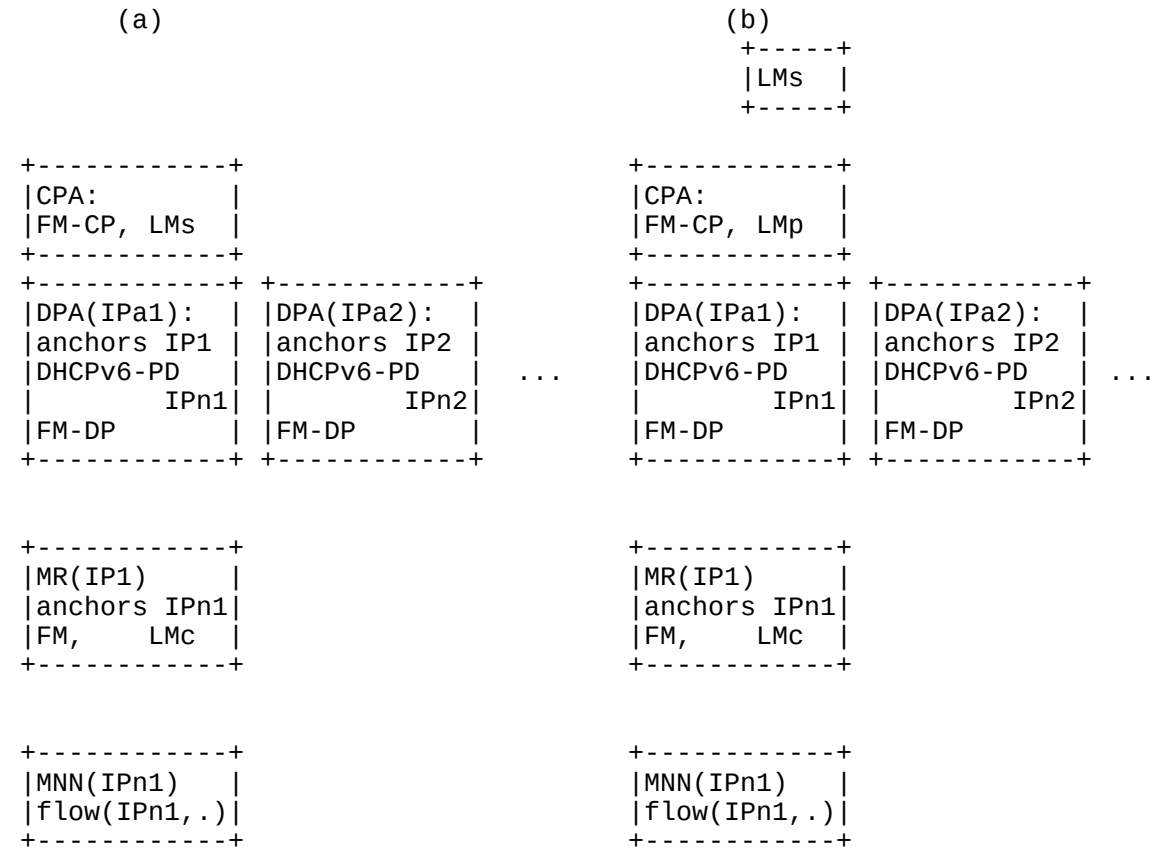


Figure 4. Configurations of NEMO basic support for a MR. (a) FM-CP and LMs at CPA, FM-DP at DPA, FM and LMc at MR; (b) Separate LMs, FM-CP and LMp at CPA, FM-DP at DPA, FM and LMc at MR.

Figure 4 shows 2 configurations of host-based mobility management for a MR with multiple instances of DPA for a distributed mobility anchoring environment.

There is an FM-DP function at each of the distributed DPA.

The control plane may either be distributed (not shown) or centralized.

When the CPA co-locates with the distributed DPA there will be multiple instances of the co-located CPA and DPA (not shown).

There is FM-CP function at the CPA.

The MR possesses the mobility functions FM and LMc.

MR is allocated an IP prefix/address IP1 which is anchored to the DPA with the IP prefix/address IPa1.

A mobile network node (MNN) in the mobile network is allocated an IP prefix/address IPn1 which is anchored to the MR with the IP prefix/address IP1.

The MNN is using IPn1 to communicate with a correspondent node (CN) not shown in the figure. The flow of this communication session is shown as flow(IPn1, ...) which uses IPn1 and other parameters.

In Figure 4(a), LMs and FM-CP co-locate at the CPA.

The LMs may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

Figure 4(b) differs from Figure 4(a) in that the LMs is separated out and the proxy LMp is added between the LMs and LMc.

LMp and FM-CP co-locate at the CPA.

The LMs may be centralized whereas the LMp may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

3.2. Operations and Parameters

The operations of distributed mobility anchoring are defined in order that they may work together in expected manners 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 the operations as 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 co-locate 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 may co-locate with LMc at CPA in a flat network with network-based mobility as shown in Figure 1(a) in Section 3.1.1.

LM-cfg:2 LMs may be a separate server whereas LMc is implemented in CPA in a flat network with network-based mobility as shown in Figure 1(b) in Section 3.1.1.

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

LM-cfg:4 LMs may be a separate server with LMp implemented at CPA whereas LMc is implemented at CPN in a hierarchical network with network-based mobility as shown in Figure 2(b) in Section 3.1.2, at MN for host-based mobility as shown in Figure 3(b) in Section 3.1.3, or at MR for network mobility as shown in Figure 4(b) 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 parameters indicated above are only the minimal. 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(a) and Figure 1(b) 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(a) and Figure 2(b) 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(a) and Figure 3(b) 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(a) and Figure 4(b) 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 a 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 With forwarding table updates, changes to the forwarding table are needed at each of the affected forwarding switches in order to change the forwarding path of the packets for the flow from that originally between the CN and the home network anchor or previous AR to that between the CN and the new AR.

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 Forwarding table updates may be triggered using DHCPv6-PD prefix delegation to change the role of IP anchoring from the home network anchor or previous AR (with FM-DP) to the new anchor (with FM-DP) to which the MN is currently attached. The new anchor will then advertise routes for the delegated prefix.

With a distributed routing protocol, the updates spread out from neighbors to neighbors and will affect all the forwarding switches such that the packets sent from "any" node to MN will go to the new AR.

FM-path-tbl:6 Forwarding table updates may also be achieved through BGP update as described in [I-D.templin-aerolink], [I-D.mccann-dmm-flatarch] and also for 3GPP Evolved Packet Core (EPC) network in [I-D.matsushima-stateless-uplane-vepc] when the scope and response time can be managed.

FM-path-tbl:7 Alternatively, 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:8 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 central plane possessing a global view of all the forwarding switches.

FM-path-tbl:9 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 using DHCPv6-PD, the forwarding paths will be reverted upon prefix lease expiration.

FM-path-ind:10 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:11 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:12 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 appropirate 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 appropirate 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 overlaps. Then the new forwarding path will resemble the direct IPv6 path from the CN to the MN.

FM-DPA-tbl:5 One method to support IP mobility is through forwarding table changes triggered using DHCPv6-PD to change the role of IP anchoring from the home network anchor (DPA with FM-DP) to the new anchor (DPA with FM-DP). It therefore puts the near end of the path change at the new DPA. Forwarding table updates will subsequently propagate upstream from this DPA up to a far end DPA where the original path and the direct IPv6 path overlap.

When that far end is too far upstream the signaling of forwarding table updates may become excessive. An alternative is to use indirection (see FM-DPA-ind) from that far end to the new DPA at the near end.

Still another alternative is to combine forwarding table update with indirection.

FM-DPA-tbl:6 Changes made by FM to the forwarding tables, 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-DPA-ind:7 An alternative mobility support is 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:8 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 path.

FM-DPA-ind:9 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-state:1 In addition to the above, a flow/session may contain states with the required information for QoS, charging, etc. as needed. These states need to be transferred from the old anchor to the new anchor.

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 obtains IP prefixes from the MR.

FM-mr:2 When the MR moves from a previous AR to a new AR, the MNNs moves with the MR. Network mobility support for these MNNs may be provided by forwarding table updates such that packets destined to the MNNs will be forwarded towards the new AR instead of towards the old AR.

Changes to forwarding table entries may occur at the new AR, the aggregate router, and other affected switch/routers such that packets destined to the MNNs will be forwarded to the new AR.

Meanwhile, changes to forwarding table entries may also occur at the old AR, the aggregate router, and other affected switch/routers such that packets destined to the MNNs will not be forwarded to the old AR.

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), it is allocated an IP prefix from the attached network. 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.

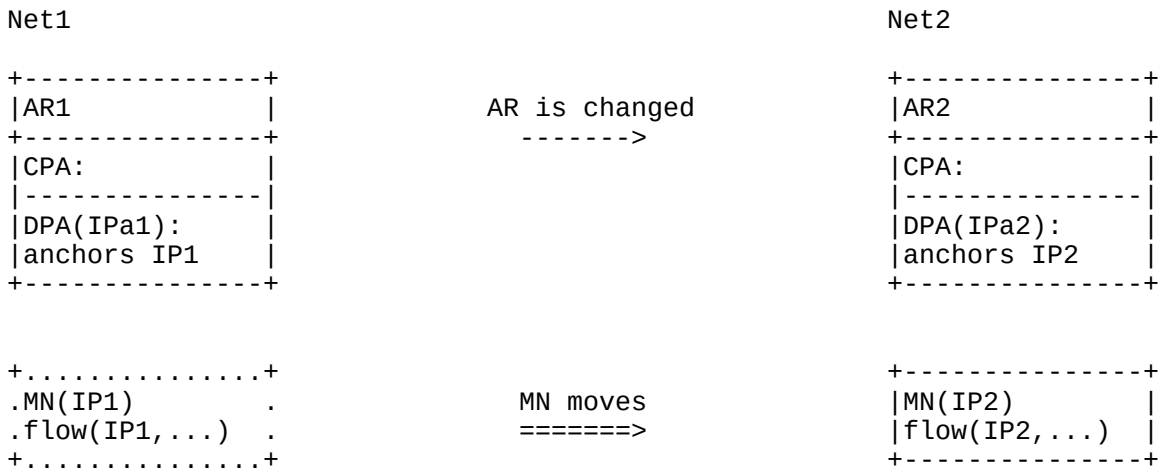


Figure 5. Changing to the new IP prefix/address. MN running a flow using IP1 in a network Net1 changes to running a flow using IP2 in Net2.

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 allocated 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.

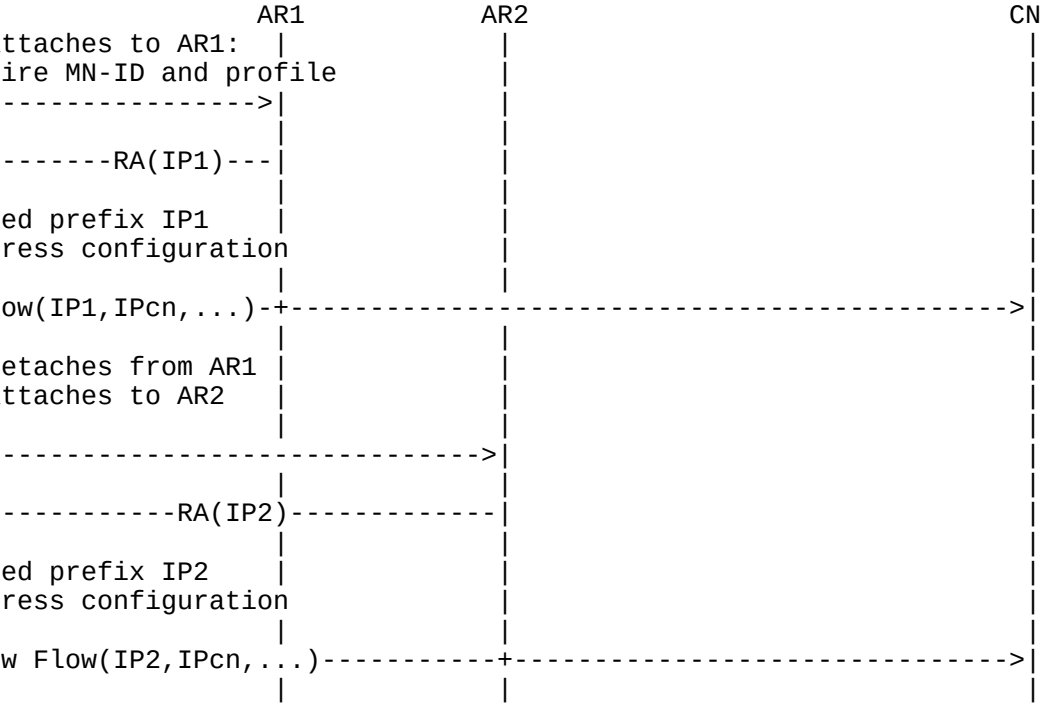


Figure 6. Re-starting a flow to use the IP allocated from the network at which the MN is attached.

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

A network or network slice may not need IP mobility support. For example, a network slice 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 or network slice 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 or network slice supporting a mix of flows requiring and not requiring IP mobility support will still need the mobility functions, which it will invoke or not invoke as needed.

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

GL-cfg:1 A network or network slice supporting a mix of flows requiring and not requiring mobility support may take any of the configurations described in Section 3.1 and need to implement in 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 used. Yet these mobility functions must not be invoked for a flow that does not need IP mobility support. 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 was 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.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.

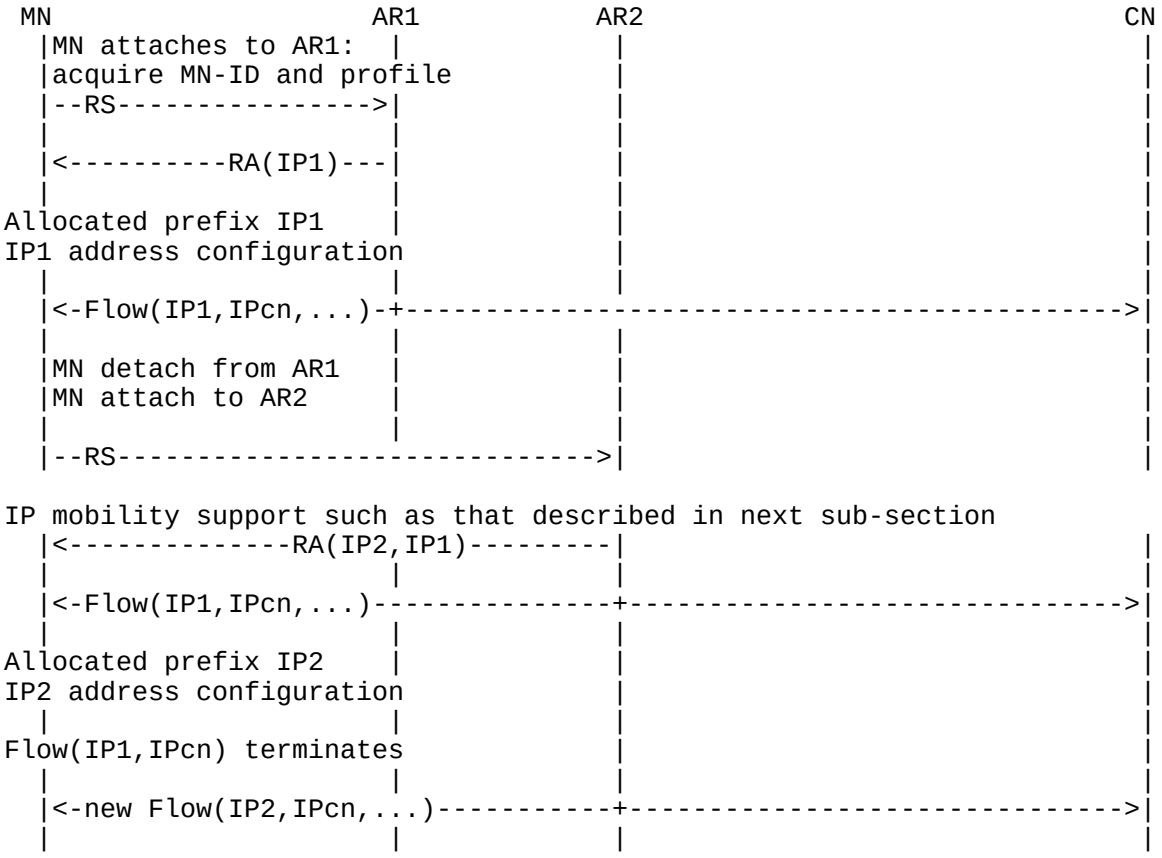


Figure 7. A flow continues to use the IP from its home network after MN has moved to a new network.

4.2.1. Guidelines for IPv6 Nodes: Need of IP Mobility

The configuration guidelines of distributed mobility for the IPv6 nodes in a network or network slice supporting a mix of flows 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 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) are to be implemented 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 or network slice supporting a mix of flows 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, FM-DPA, and FM-DPA-tbl 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 between AR2, the far end DPA and other affected routers so that these packets will traverse towards AR2. Meanwhile, changes to the forwarding table entries will also occur between AR1, the far end DPA and other affected routers so that if these packets ever reaches any of them, the they will not traverse towards AR1 but will traverse towards AR2. 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 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 Figures 1(a) and 1(b) in Section 3.1 are implemented as shown in Figure 8.

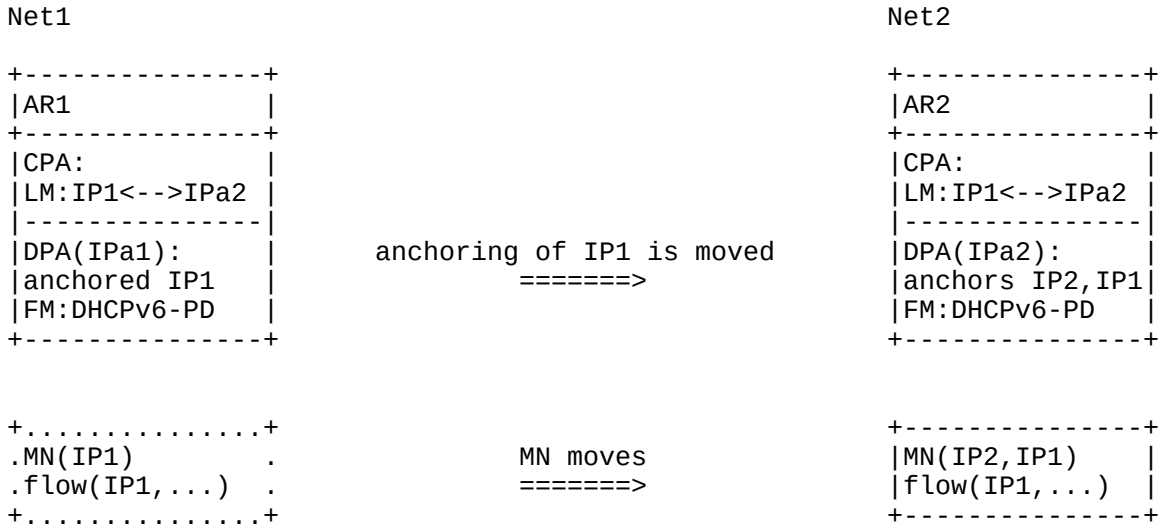


Figure 8. IP prefix/address anchor switching to the new network. MN with flow using IP1 in Net1 continues to run the flow using IP1 as it moves to Net2.

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. BGP UPDATE messages may be used to change the forwarding table entries as described in [I-D.templin-aerolink] and [I-D.mccann-dmm-flatarch] if the response time of such updates does not exceed the handover delay requirement of the flow. An alternative 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 or network slice supporting a mix of flows 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(a) or Figure 1(b) in Section 3.1 for a flat network.

The appropriate IPv6 nodes (CPA, DPA) are to be implemented 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 or network slice supporting a mix of flows 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 FM functions are implemented through the DHCPv6-PD protocol. Here the anchor operations to properly forward the packets for a flow as described in the FM operations and mobility message parameters in FM-path, FM-path-tbl, FM-DPA, and FM-DPA-tbl in Section 3.2.2 are realized by changing the anchor with DHCPv6-PD and also by reverting such changes later after the application has already closed and when the DHCPv6-PD timer expires. 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 management function in the anchor node at a new network must allow to assign 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 management function in the anchor node must allow 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 management function in the mobile node must allow 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. The security management function in the mobile node also allows 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 Figures 1(a) and 1(b) in Section 3.1 for which the FM-CP and the LM are centralized and the FM-DP's 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 the DHCPv6-PD protocol.

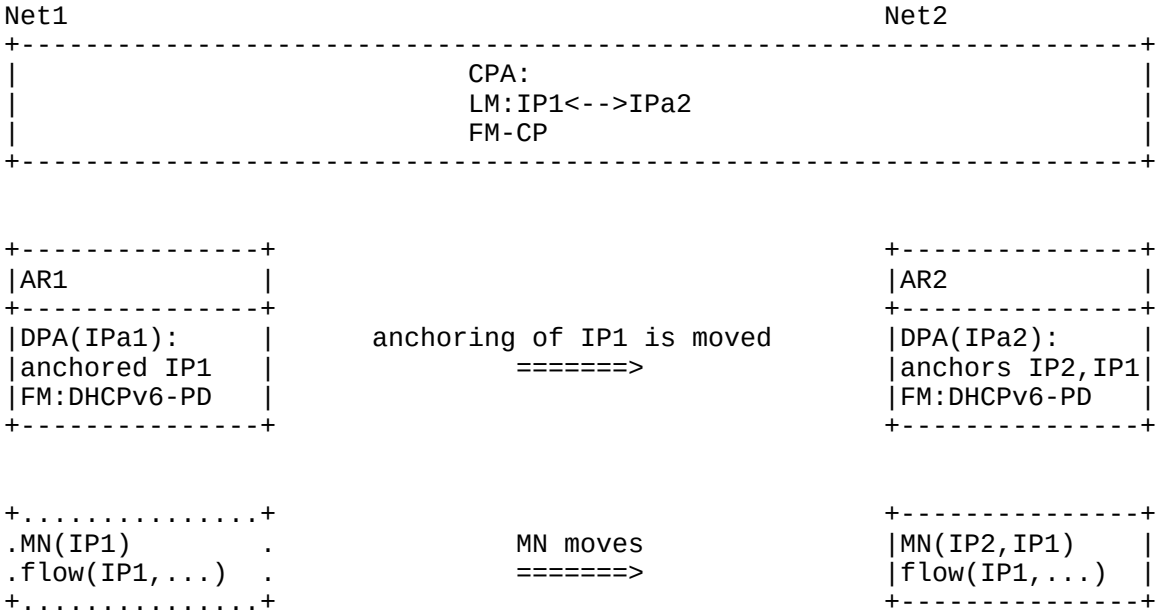


Figure 9. IP prefix/address anchor switching to the new network with with the LM and the FM-CP in a centralized control plane whereas the FM-DP's are distributed.

The example call flow in Figure 10 shows that MN is allocated IP1 when it 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 allocated from the new network.

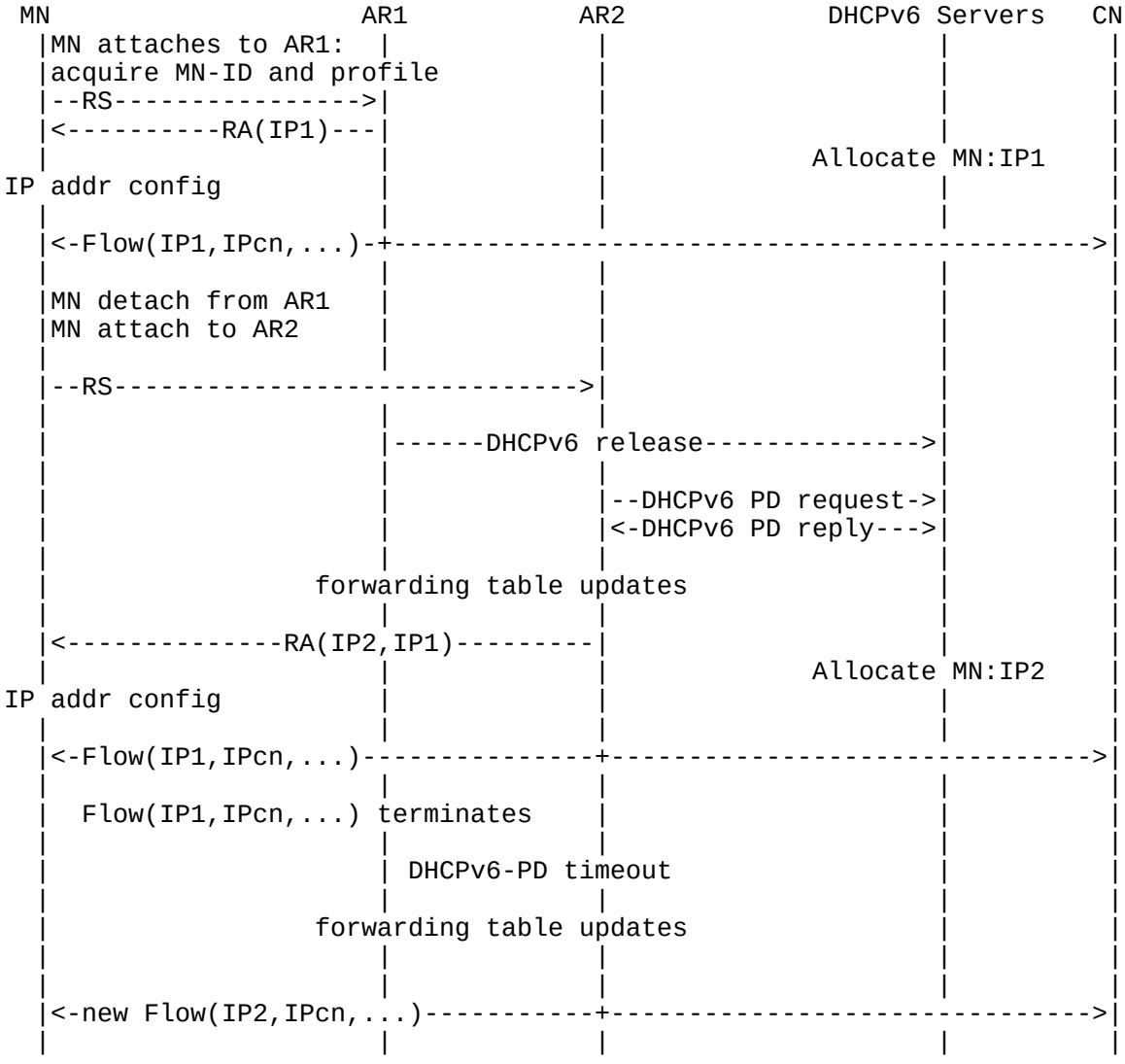


Figure 10. DMM solution. MN with flow using IP1 in Net1 continues to run the flow using IP1 as it moves to Net2.

As the MN moves from AR1 to AR2, the AR1 as a DHCPv6 client may send a DHCPv6 release message 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 allocate 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].

Knowing that MN is using IP1, the AR2 sends to a DHCPv6 server a DHCPv6-PD request to move the IP1 to AR2. The server sends to AR2 a DHCPv6-PD reply to move the IP1. 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 AR2, with prefix information that includes IP1 and IP2. The MN may then continue to use IP1. In addition, the MN is allocated the prefix IP2 with which it may configure its IP addresses. Now for flows using IP1, packets destined to IP1 will be forwarded to the MN via AR2.

As such flows have terminated and DHCPv6-PD has timed out, 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 or network slice with centralized control plane and supporting a mix of flows 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(a) or Figure 1(b) in Section 3.1 with centralized control plane for a flat network.

The appropriate IPv6 nodes (CPA, DPA) are to be implemented 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 or network slice supporting a mix of flows 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 as described in the FM operations and mobility message parameters in FM-path, FM-path-tbl, FM-DPA, and FM-DPA-tbl in Section 3.2.2 is realized by changing the anchoring with DHCPv6-PD and undoing such changes later when its timer expires and 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 Figures 2(a) and 2(b) 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 (FW's). 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 allocated 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.

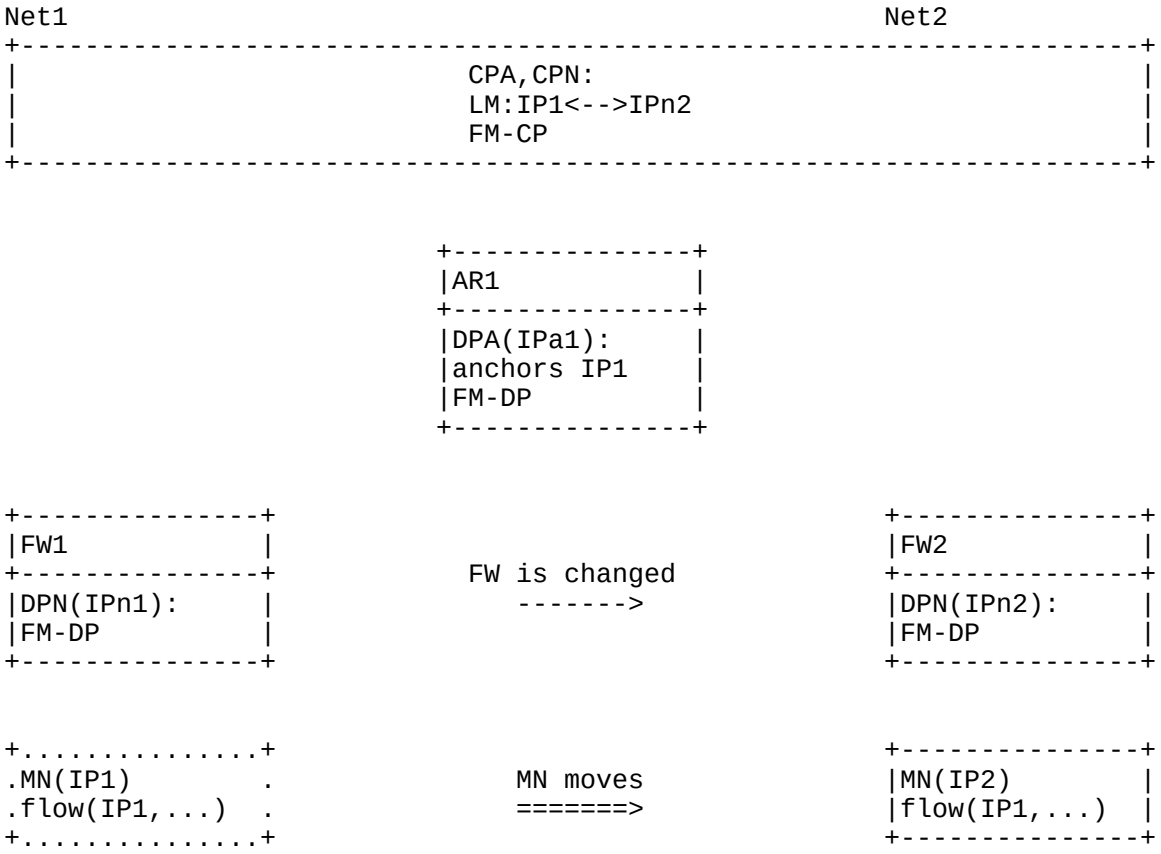


Figure 11. Mobility without involving change of IP anchoring in a network in which the IP prefix allocated to the MN is anchored at an AR which is hierarchically above multiple FWs to which the MN may connect.

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

The configuration guideline for a hierarchical network or network slice with centralized control plane and supporting a mix of flows 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(a) or Figure 2(b)in Section 3.1.2 with centralized control plane for a hierarchical network.

The appropriate IPv6 nodes (CPA, DPA) are to be implemented 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 or network slice supporting a mix of flows 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 provides 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 Figures 2(a) and 2(b) 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 (FW's).

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.

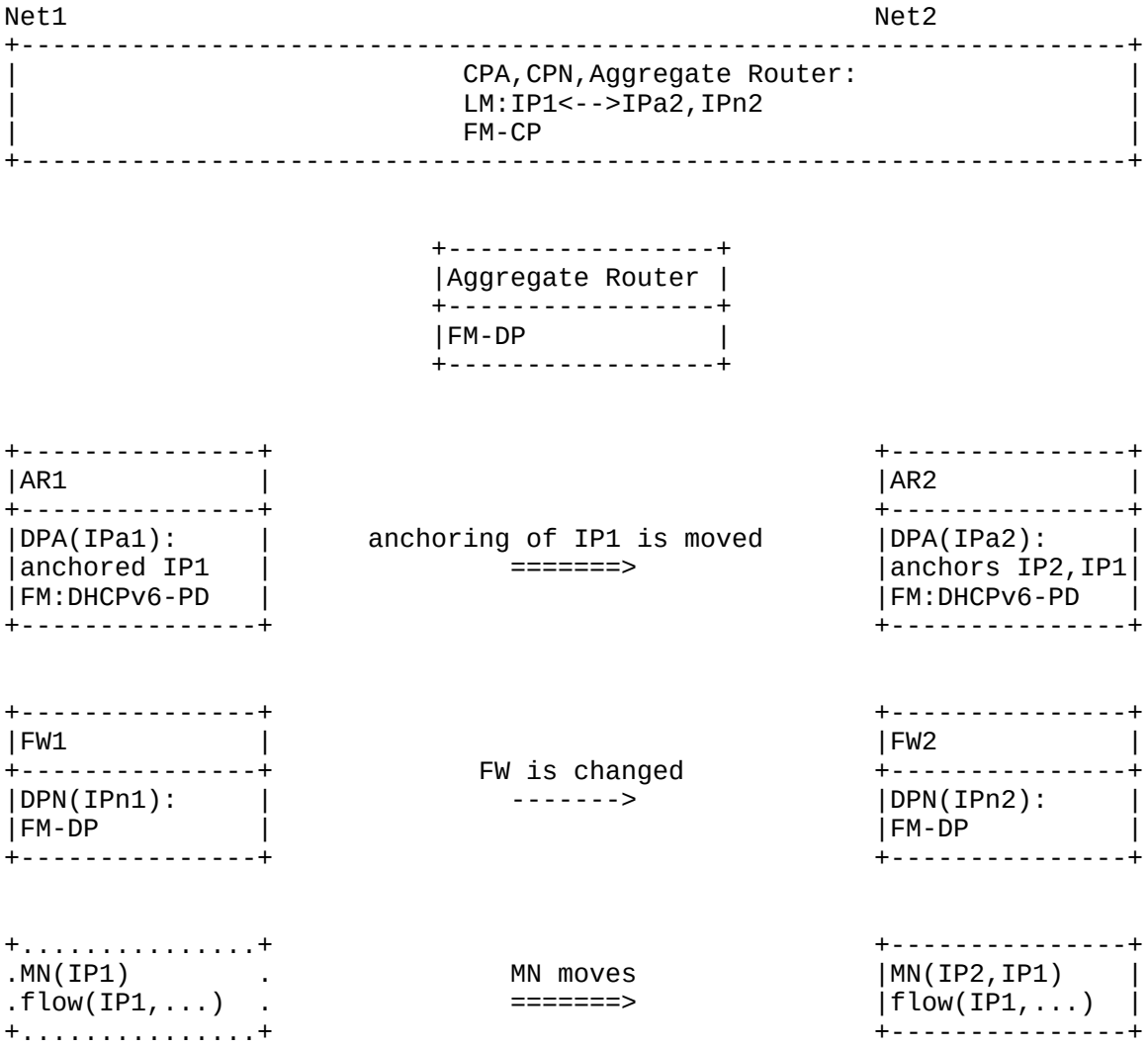


Figure 12. Mobility involving change of IP anchoring in a network with hierarchy in which the IP prefix allocated to the MN is anchored at an Edge Router supporting multiple access routers to which the MN may connect.

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.

To change the anchoring of IP1, AR1 acting as a DHCPv6-PD client may exchange message with the DHCPv6 server to release the prefix IP1. Meanwhile, AR2 acting as a DHCPv6-PD client may exchange message with the DHCPv6 server to delegate the prefix IP1 to AR2.

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

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

The GL-mix guidelines in Section 4.1.1 and in Section 4.2.1 for the IPv6 nodes for a network or network slice supporting a mix of flows 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 Figures 4(a) and 4(b) 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 a MNN is attached and is allocated with an IP address from IPn1.

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 and therefore moves with the MR.

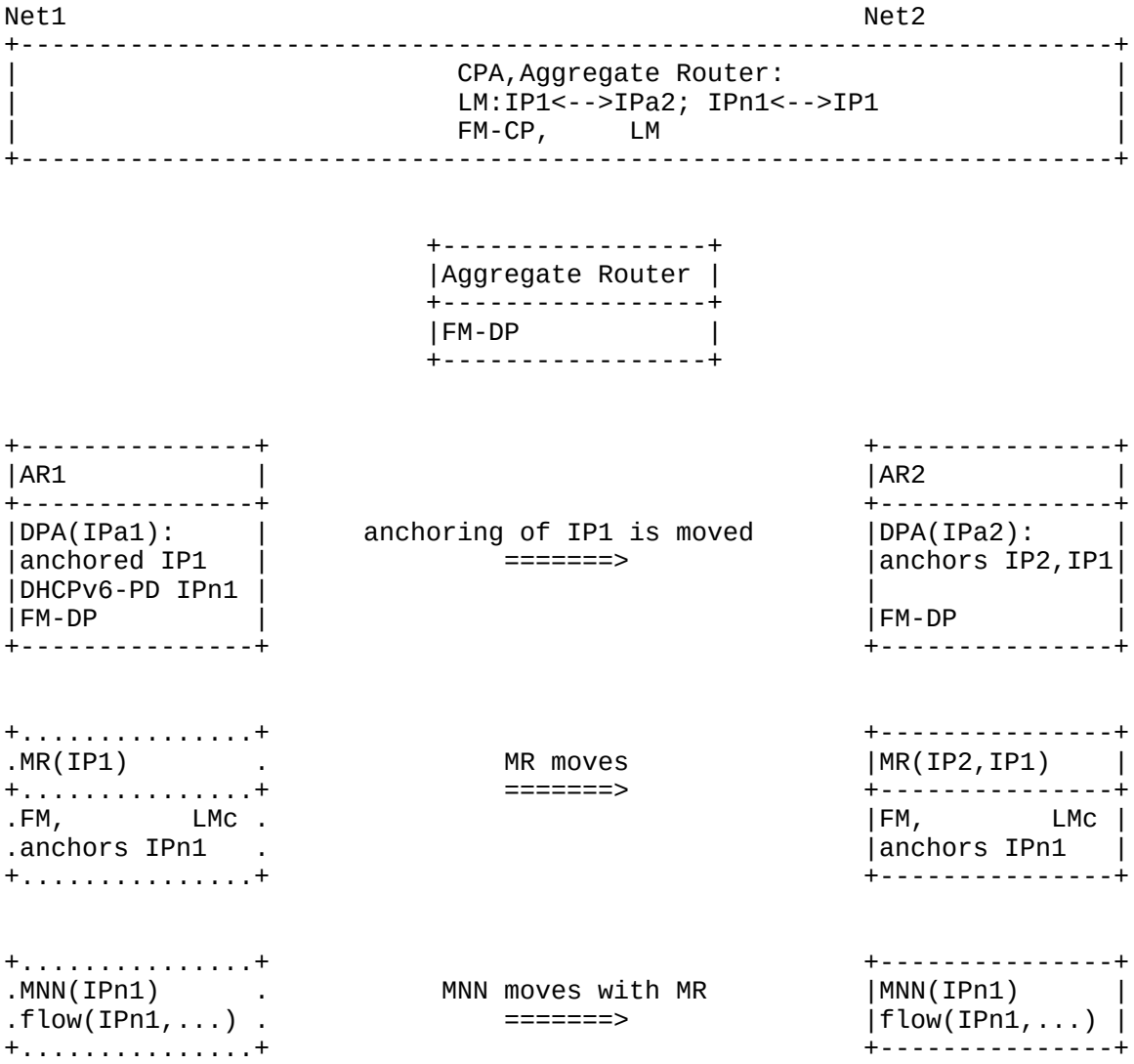


Figure 13. Mobility involving change of IP anchoring for a MR to which a MNN is attached.

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 or network slice 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(a) or Figure 4(b) in Section 3.1.

The appropriate IPv6 nodes (CPA, DPA) are to be implemented 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 or network slice supporting a mix of flows 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 AR, so that the MR may support potential MNN 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 between AR2, the aggregate router and other affected routers 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 between AR1, the aggregate router and other affected routers so that such packets ever reaches any of them, the packet will not traverse towards AR1 but will traverse towards AR2.

GL-switch:11 The security management function in the anchor node at a new network must allow to assign the original IP prefix/address allocated to the MR and used by the MNN at the previous (original) network. As the assigned original IP prefix/address is to be used in the new network, the security management function in the anchor node must allow 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 management function in the mobile router must allow 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 management function in the mobile router also allows to use the original IP address by the MNNs for the previous flow in the new network.

6. Security Considerations

The security considerations are already described in different sessions through this document. They are described in terms of integrity support, privacy support etc. in describing the mobility functions in Section 3.2. They are also described in the guidelines for IPv6 nodes in various subsections Section 4 and Section 5.

7. IANA Considerations

This document presents no IANA considerations.

8. Contributors

This document has benefited from other work on mobility solutions using BGP update, on mobility support in SDN network, on providing mobility support only when needed, and on mobility support in enterprise network. These work 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, Peter McCann, Pierrick Seite, Jouni Korhonen, and Sri Gundavelli.

Valuable comments have also been received from John Kaippallimalil, ChunShan Xiong, and Dapeng Liu.

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-00 (work in progress), August 2016.

[I-D.ietf-dmm-fpc-cpdp]
Matsushima, S., Bertz, L., Liebsch, M., Gundavelli, S., and D. Moses, "Protocol for Forwarding Policy Configuration (FPC) in DMM", draft-ietf-dmm-fpc-cpdp-05 (work in progress), October 2016.

[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-09 (work in progress), December 2016.

[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-flatarch]
McCann, P., "Authentication and Mobility Management in a Flat Architecture", draft-mccann-dmm-flatarch-00 (work in progress), March 2012.

[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.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-74 (work in progress), November 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>.

[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: September 14, 2017

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

March 13, 2017

Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-07

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

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

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

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

This Internet-Draft will expire on September 14, 2017.

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.

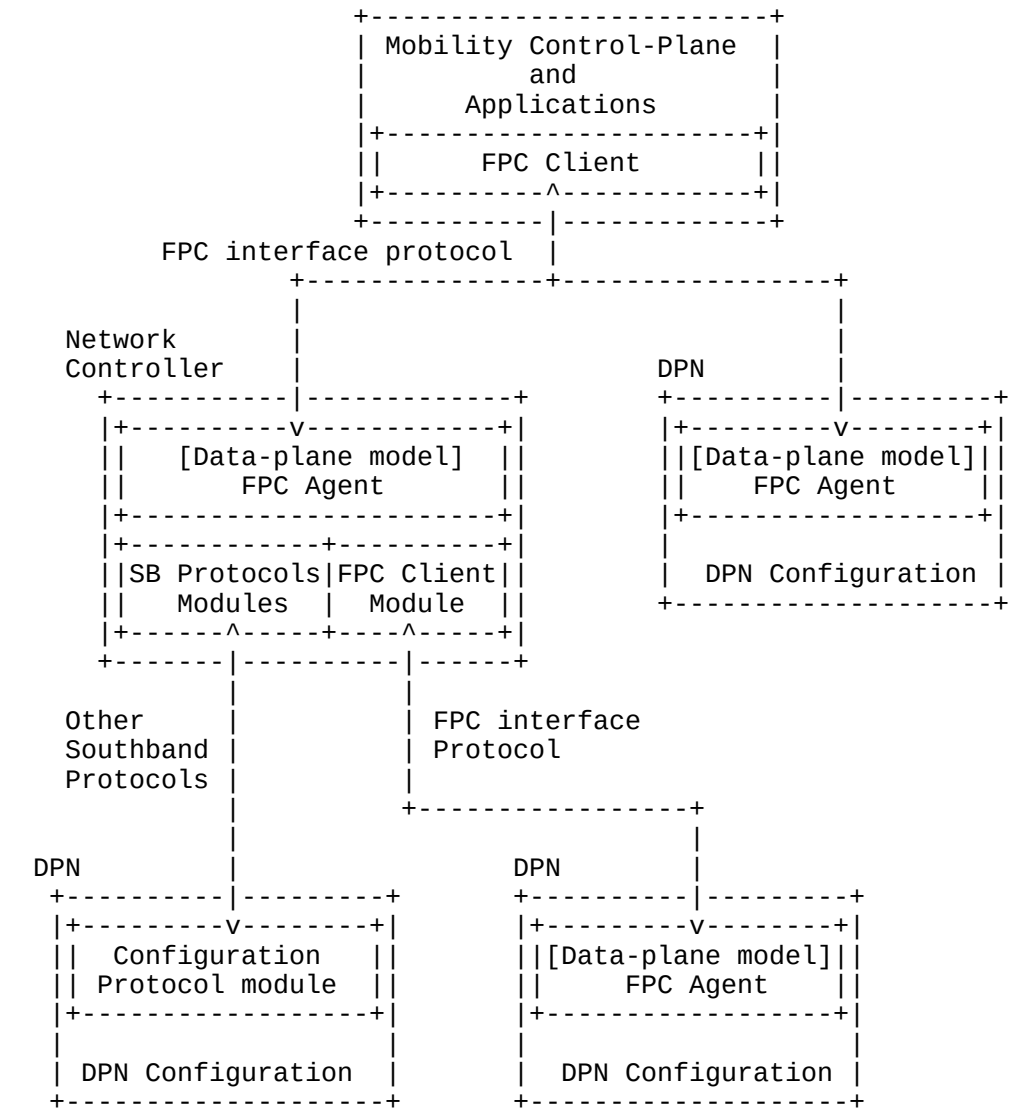


Figure 1: Reference Forwarding Policy Configuration (FPC) Architecture

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.

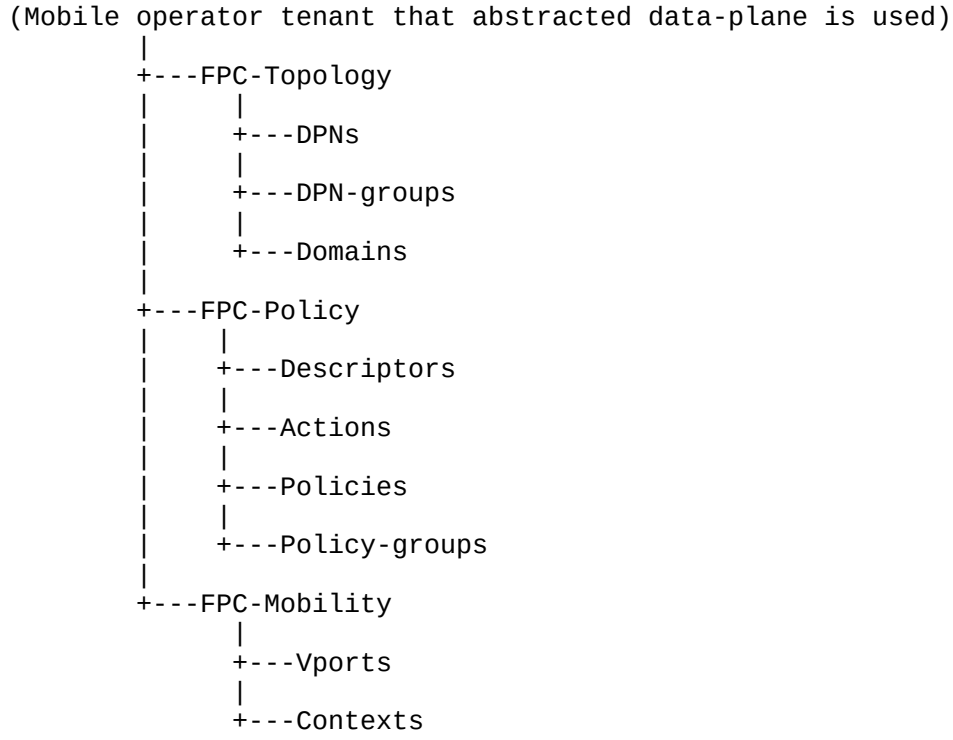


Figure 2: FPC Data-plane Information Model

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.

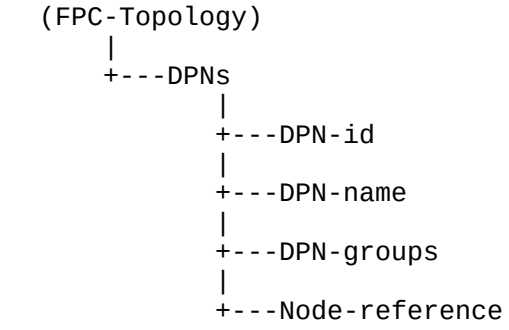


Figure 3: DPNs Model Structure

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.

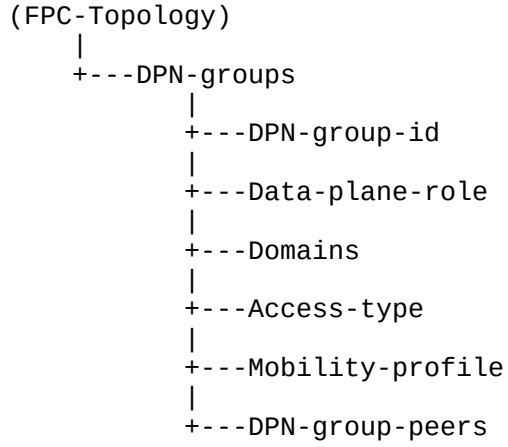


Figure 4: DPN-groups Model Structure

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.

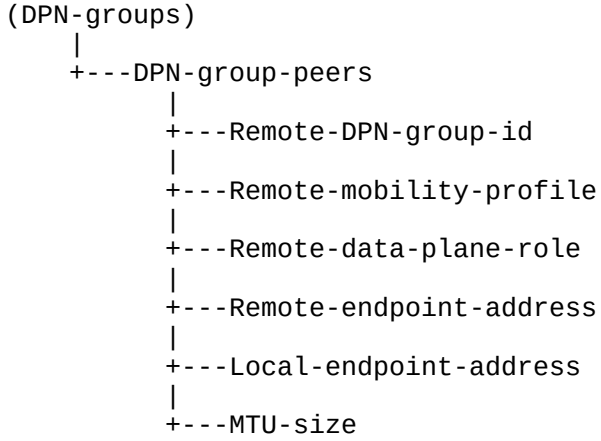


Figure 5: DPN-groups Peer Model Structure

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.

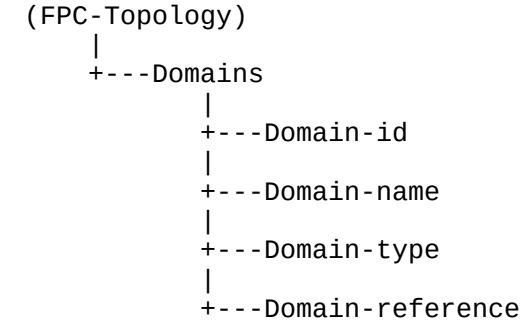


Figure 6: Domain Model Structure

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.

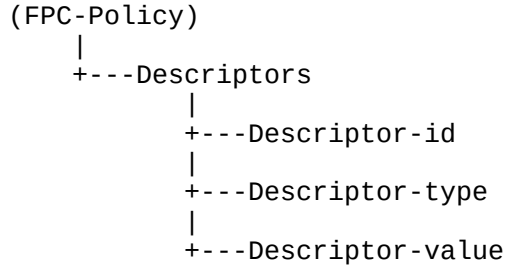


Figure 7: Descriptor Model Structure

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.

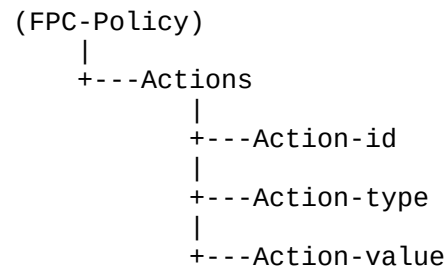


Figure 8: Action Model Structure

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).

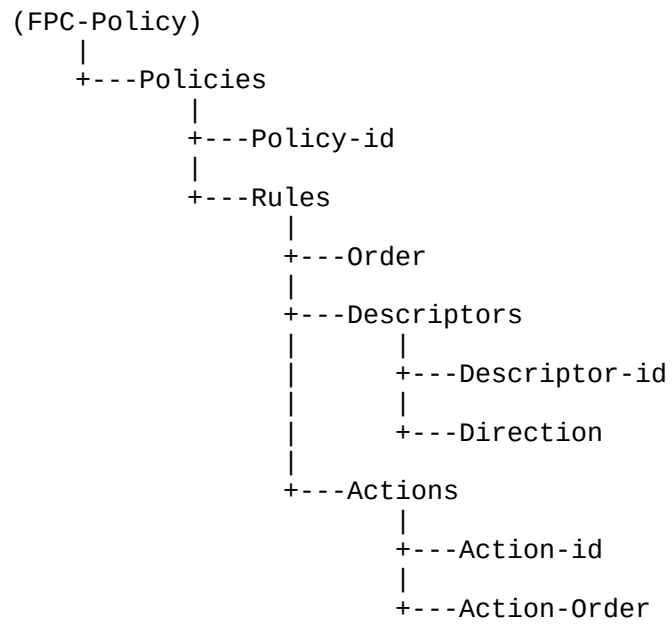


Figure 9: Model Structure for Policies

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.

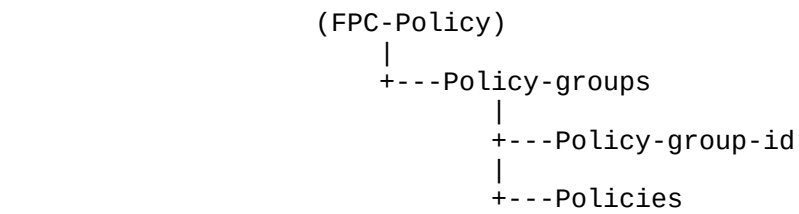


Figure 10: Policy-group Model Structure

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.

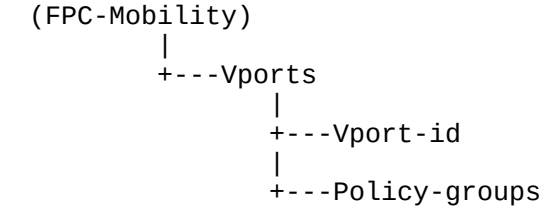


Figure 11: Vport Model Structure

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.

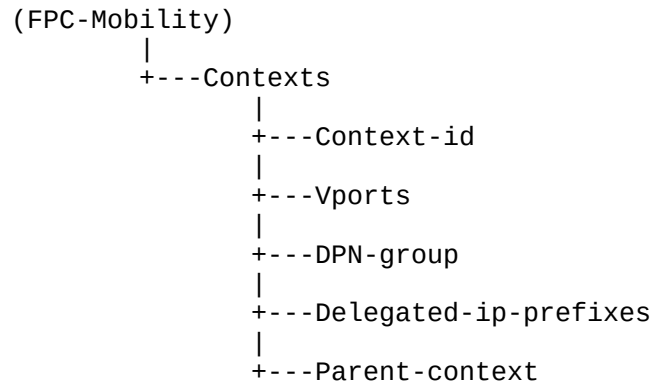


Figure 12: Common Context Model Structure

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.

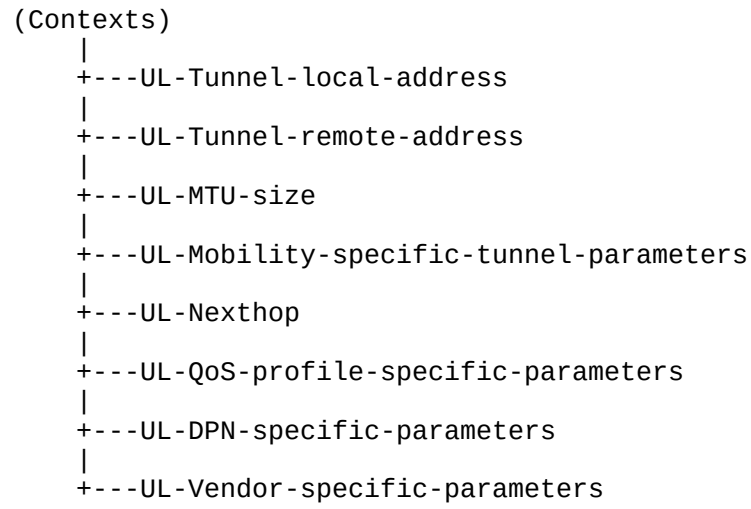


Figure 13: Uplink Context Model of Single DPN Structure

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.

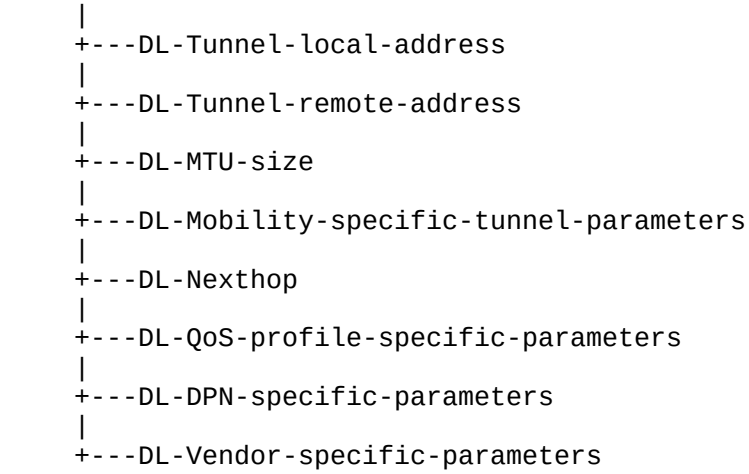


Figure 14: Downlink Context Model of Single DPN Structure

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'.

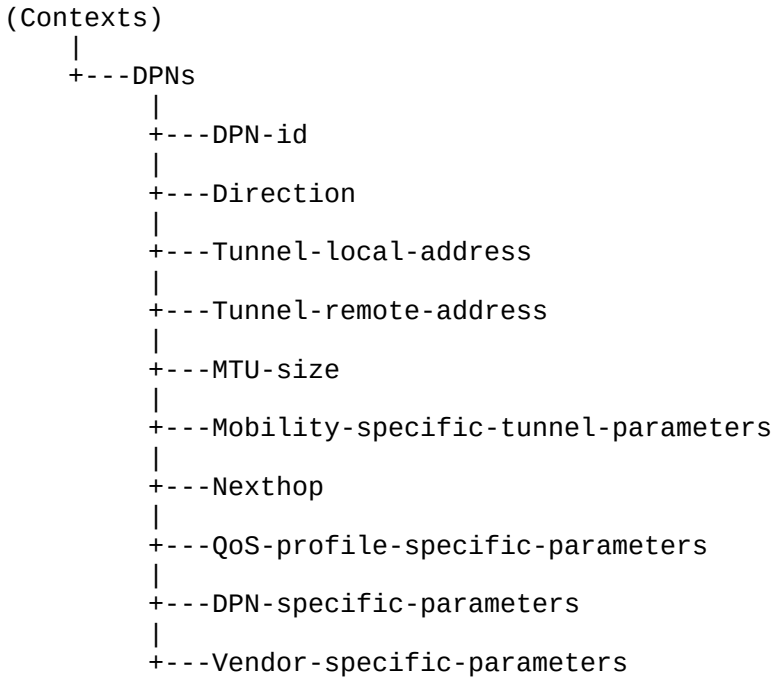


Figure 15: Multiple-DPN Supported Context Model Structure

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.

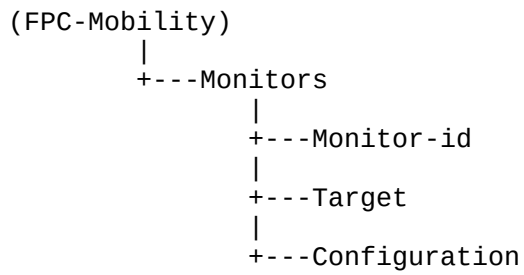


Figure 16: Common Monitor Model Structure

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

- none - all entities have no references to other entities. This implies only Contexts are present. Vports MUST have references to Policy-Groups.
- op - All references are contained in the operation body, i.e. only intra-operation references exist.
- 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'.
- storage - One or more references exist outside of the operation and bundle. A lookup to a cache / storage is required.
- 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.

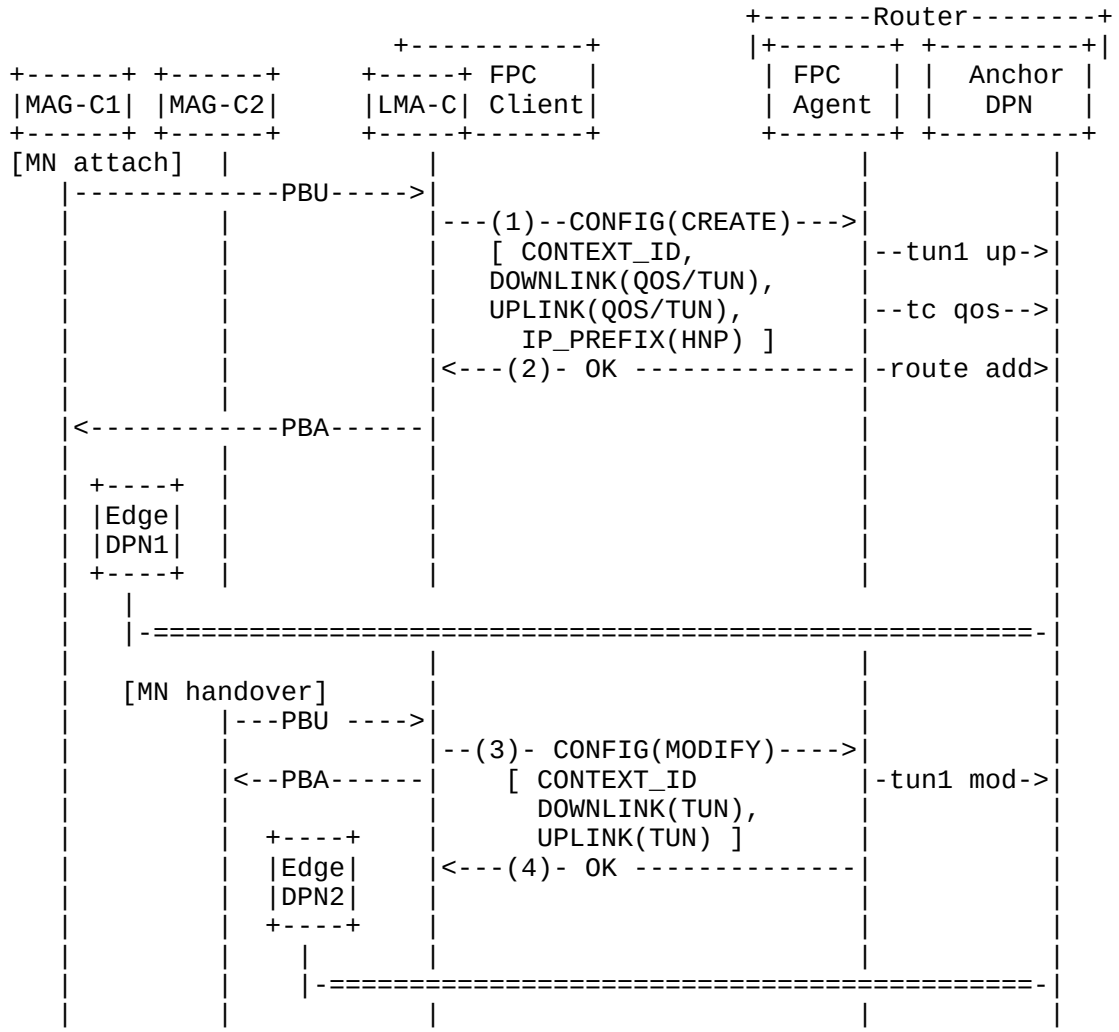


Figure 17: Exemplary Message Sequence (focus on FPC reference point)

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).

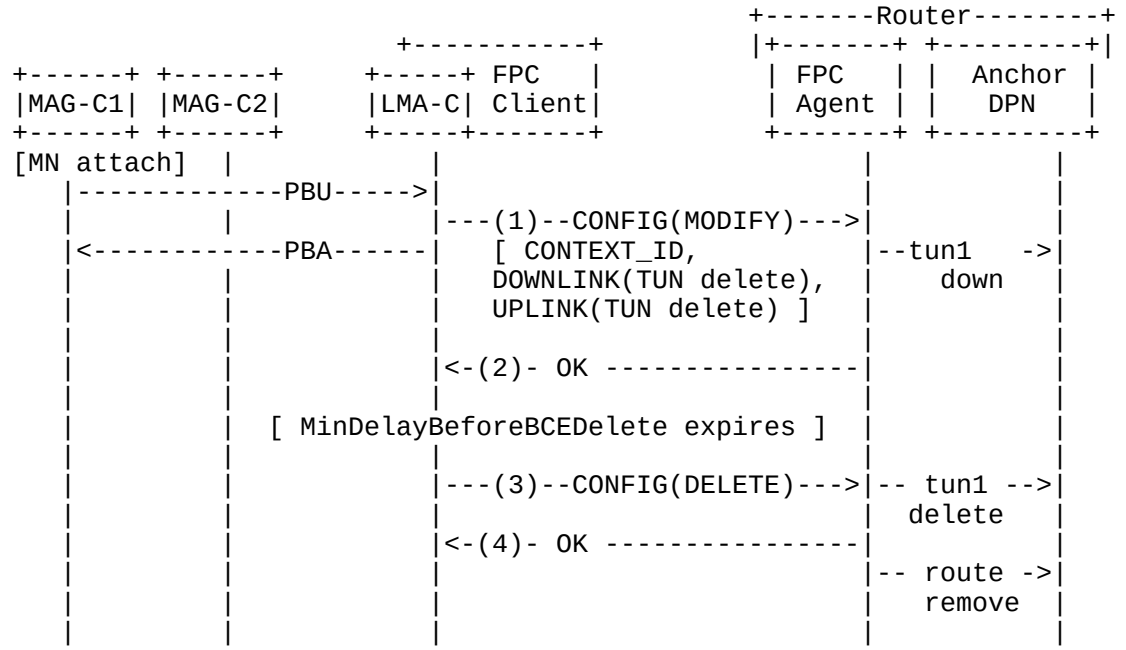


Figure 18: Exemplary Message Sequence (focus on FPC reference point)

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.

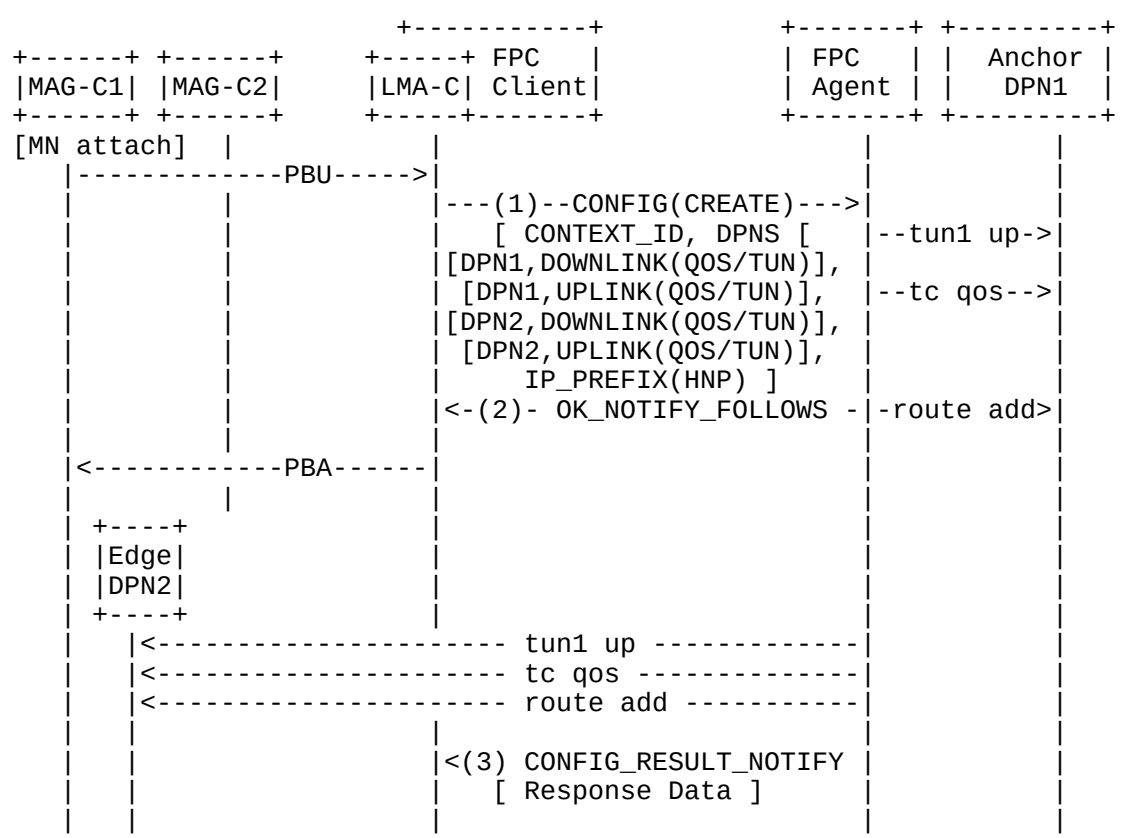


Figure 19: Exemplary Message Sequence for Multi-DPN Agent

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.

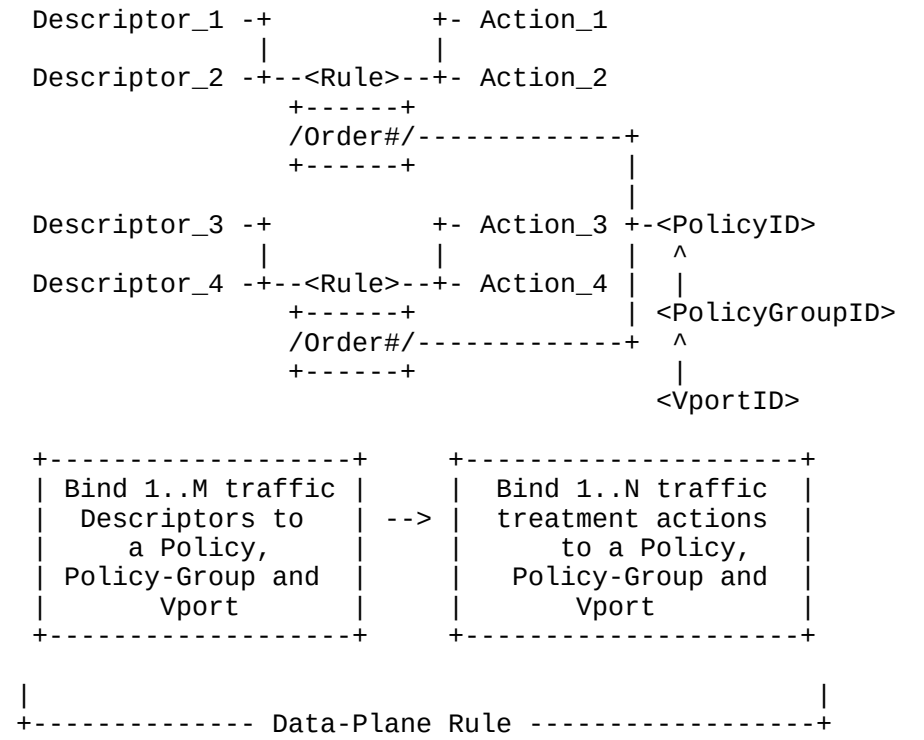


Figure 20: Structure of Policies and Vports

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.

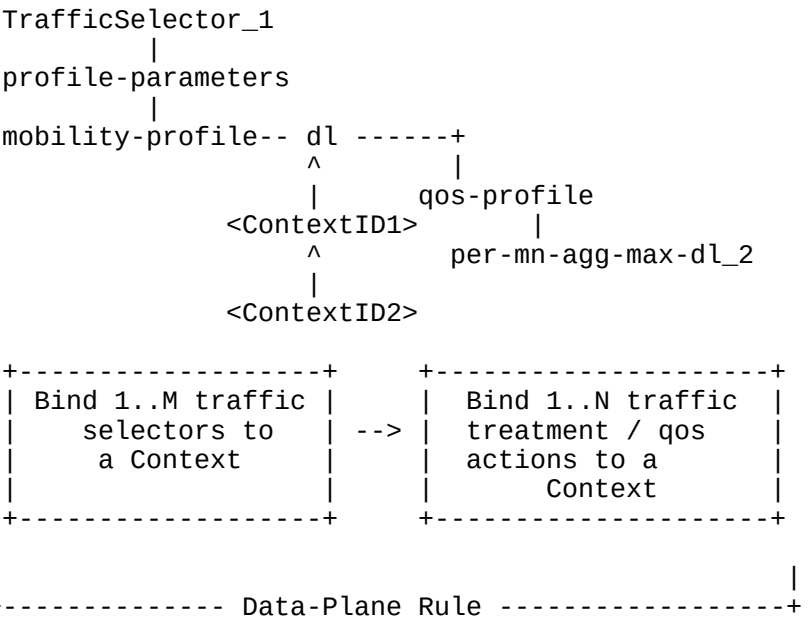


Figure 21: Structure of Contexts

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).

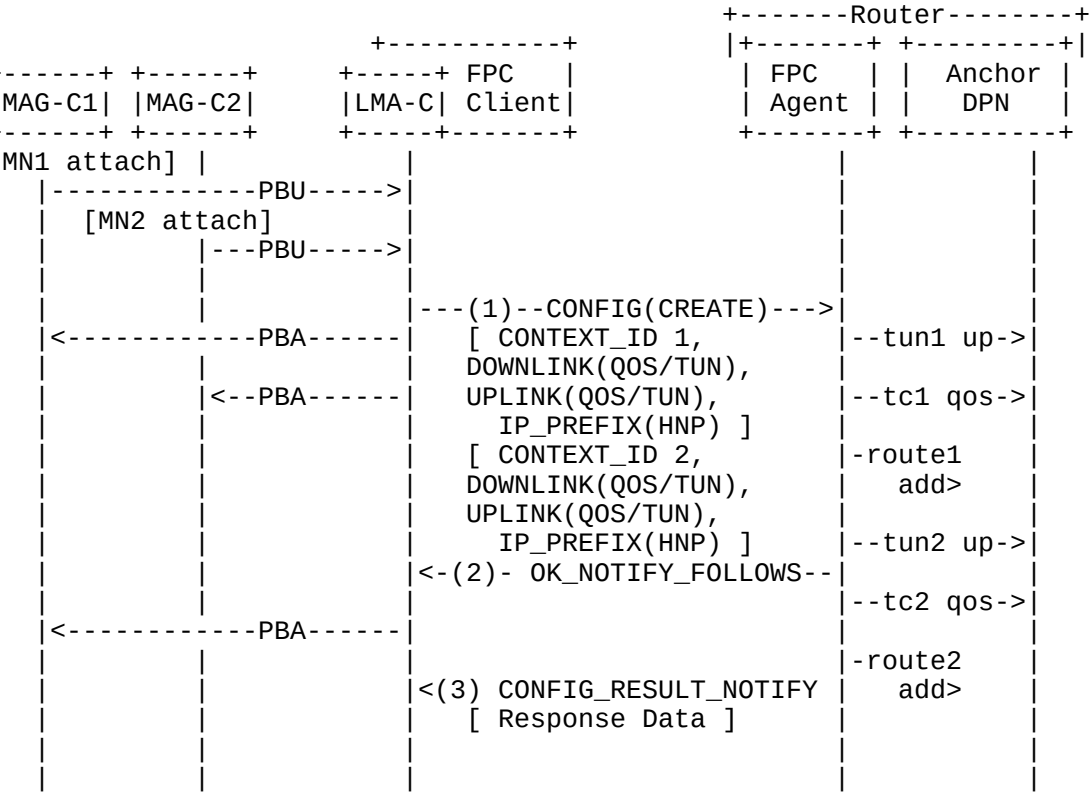


Figure 22: Exemplary Bulk Entity with Asynchronous Notification Sequence (focus on FPC reference point)

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.

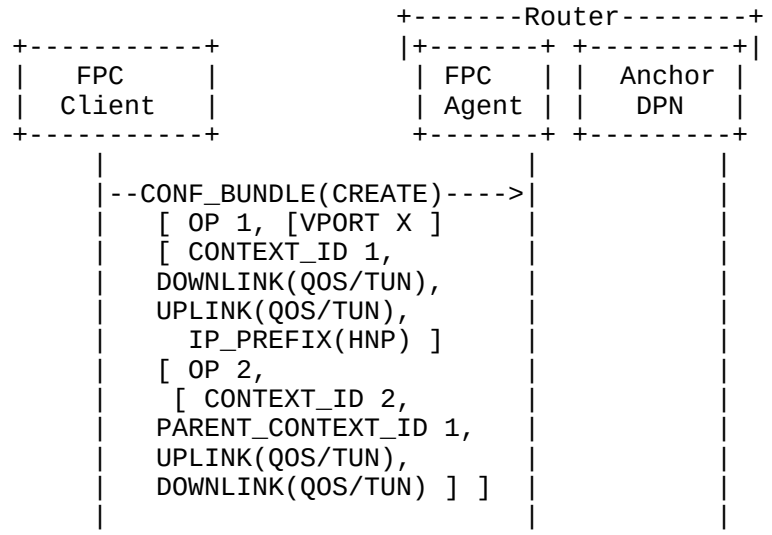


Figure 23: Exemplary Bundle Message (focus on FPC reference point)

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.

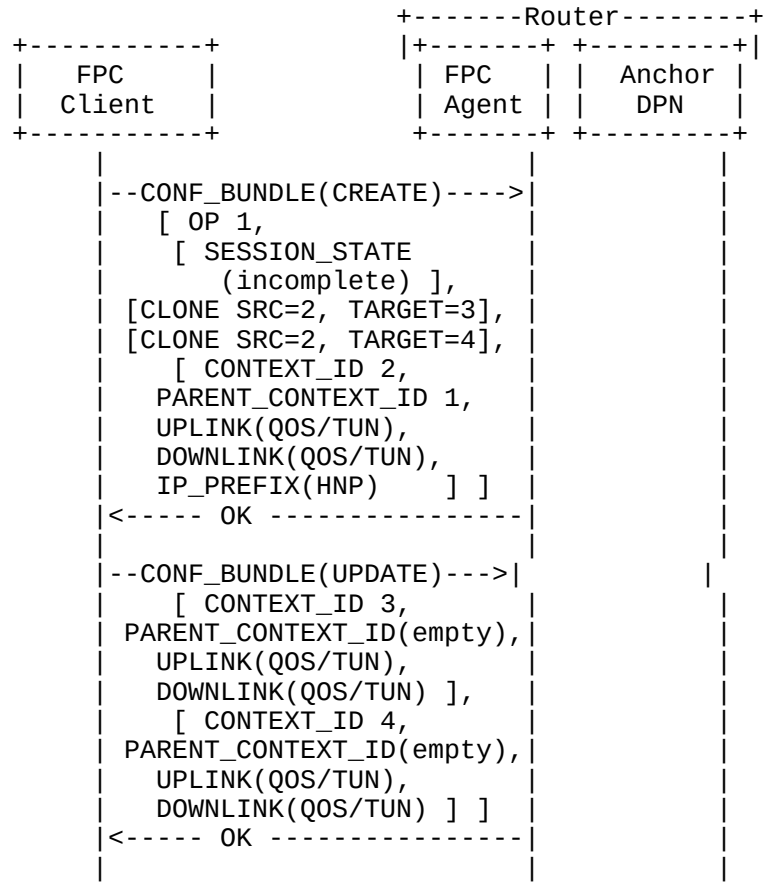


Figure 24: Exemplary Bundle Message (focus on FPC reference point)

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

- 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.

- o 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.

- o 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.

- o 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.

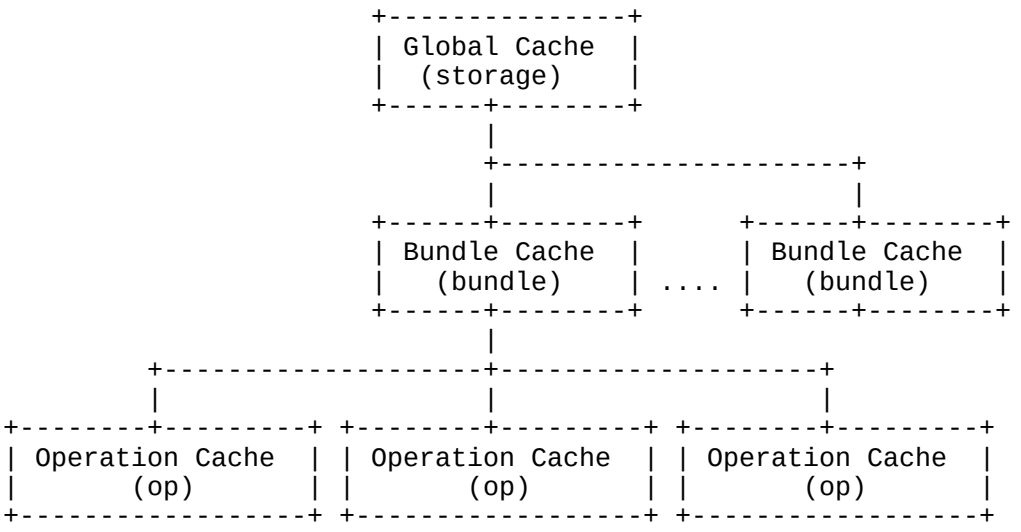


Figure 25: Exemplary Hierarchical Cache

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.

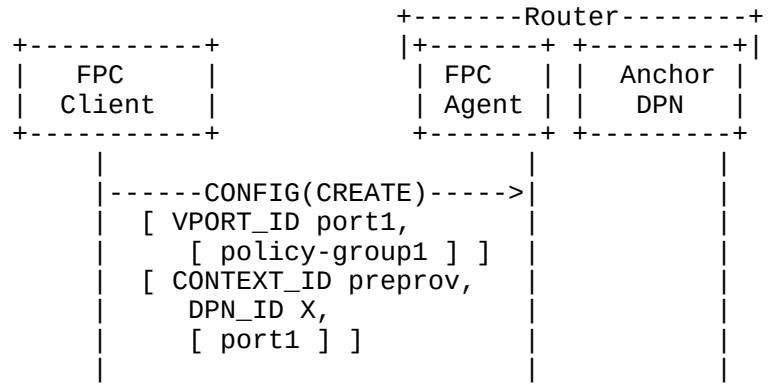


Figure 26: Exemplary Config Message for policy pre-provisioning

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:

- Events specified in the Event List of an EVENTS CONFIG
- LOW_THRESHOLD_CROSSED
- HIGH_THRESHOLD_CROSSED
- PERIODIC_REPORT
- SCHEDULED_REPORT
- PROBED
- 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.

- assign-ip - Assign the IP Address for the mobile session.
- assign-dpn - Assign the Dataplane Node.
- session - Assign values for the Session Level.
- uplink - Command applies to uplink.
- 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.

- o assign-ip - Assign the IP Address for the mobile session.
- o assign-dpn - Assign the Dataplane Node.
- 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' this implies the values are part of the default bearer.
- o uplink - Command applies to uplink.
- o 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., Field, B., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., and D. Lebrun, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-05 (work in progress), February 2017.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-12 (work in progress), February 2017.

[I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., Shakir, R., jefftant@gmail.com, j., and E. Crabbe, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-07 (work in progress), February 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>.

[RFC6088] Tsirtsis, G., Giarreta, G., Soliman, H., and N. Montavont, "Traffic Selectors for Flow Bindings", RFC 6088, DOI 10.17487/RFC6088, January 2011, <http://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, <http://www.rfc-editor.org/info/rfc6089>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://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, <http://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-03 (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-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, <http://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, <http://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, <http://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, <http://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, <http://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
Internet-Draft
Intended status: Informational
Expires: August 2, 2017

A. Yegin
Actility
D. Moses
Intel
K. Kweon
J. Lee
J. Park
Samsung
S. Jeon
Sungkyunkwan University

January 29, 2017

On Demand Mobility Management
draft-ietf-dmm-ondemand-mobility-10

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on August 2, 2017.

Table of Contents

1. Introduction

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

IP session continuity: The ability to maintain an ongoing IP session by keeping the same local end-point IP address throughout the session despite the mobile host 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 the IP session continuity. IP session continuity is essential for mobile hosts to maintain ongoing flows without any interruption.

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

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

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

Achieving IP session continuity and IP address reachability by using Mobile IP incurs some cost. Mobile IP protocol forces the mobile host’s IP traffic to traverse a centrally-located router (Home Agent, HA), which incurs additional transmission latency and use of additional network resources, adds to the network CAPEX and OPEX, and decreases the reliability of the network due to the introduction of a single point of failure [RFC7333]. Therefore, IP session continuity and IP address reachability should be be provided only when needed.

Furthermore, when an application needs session continuity, it may be able to satisfy that need by using a solution above the IP layer, such as MPTCP [RFC6824], SIP mobility [RFC3261], or an application-layer mobility solution. Those higher-layer solutions are not subject to the same issues that arise with the use of Mobile IP since they can utilize the most direct data path between the end-points. But, if Mobile IP is being applied to the mobile host, those higher-layer protocols are rendered useless because their operation is inhibited by the Mobile IP. Since Mobile IP ensures 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 the applications running on the mobile host to indicate whether they need IP session continuity or IP address reachability. The network protocol stack on the mobile host, in conjunction with the network infrastructure, would provide the required type of IP service. It is for the benefit of both the users and the network operators not to engage an extra level of service unless it is absolutely necessary. So it is expected that applications and networks compliant with this specification would utilize this solution to use network resources more efficiently.

2. Notational Conventions

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

3. Solution

3.1. Types of IP Addresses

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

- Fixed IP Address

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

Applications running as servers at a published IP address require a Fixed IP Address. Long-standing applications (e.g., an SSH session) may also require this type of address. 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.

The network creates the desired guarantee (Fixed, Session-lasting or Non-persistent) by either assigning the address prefix (as part of a stateless address generation process), or by assigning an IP address (as part of a stateful IP address generation).

The exact mechanism of prefix or address assignment is outside the scope of this specification.

3.2. Granularity of Selection

The IP address type selection is made on a per-socket granularity. Different parts of the same application may have different needs. For example, control-plane of an application may require a Fixed IP Address in order to stay reachable, whereas data-plane of the same application may be satisfied with a 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, and zero or more Fixed IP addresses may be configured on the IP stack of the host. The combination may be as a result of the host policy, application demand, or a mix of the two.

When an application requires a specific type of IP address and such 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 Section 3.4 for more details). If the operation fails, the IP stack shall fail the associated socket request. 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 selecting to use available addresses or choose 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 address (Fixed, Session-lasting or Non-persistent) and how the network indicates the type of address in its advertisement of IP prefixes or addresses (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 Selection

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

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

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

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

Unfortunately, these flags do not satisfy the aforementioned needs due to the following reasons:

- Current flags indicate a "preference" whereas there is a need for indicating "requirement". Source address selection algorithm does not have to produce an IP address compliant with the "preference" , but it has to produce an IP address compliant with the "requirement".

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

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

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

IPV6_REQUIRE_FIXED_IP /* Require a Fixed IP address as source */

IPV6_REQUIRE_SESSION_LASTING_IP /* Require a Session-lasting IP address as source */

IPV6_REQUIRE_NON_PERSISTENT_IP /* Require a Non-persistent IP address as source */

Only one of these flags may be set on the same socket. If an application attempts to set more than one flag, the most recent setting will be the one in effect.

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

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

When the IP stack is required to use a source IP address of a specified type, it can perform one of the following: It can use an existing address (if it has one), or it can create a new one from an existing prefix of the desired type. If the host does not already have an IPv6 prefix of the specific type, it can 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 since its configuration), on the other hand, acquiring a new IP prefix from the network may take some time (due to signaling exchange with the network) and may fail due to network policies.

An additional new flag - ON_NET flag - enables the application to direct 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:

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 exists. If there is no preconfigured IP address of the desired type, the IP stack will request a IPv6 prefix from the current serving network (regardless of whether this flag is set or not).

The ON_NET flag must be used together with one of the 3 flags defined above. If ON_NET flag is used without any of these flags, it must be

ignored. If the ON_NET flag is not used, the IP stack is free to either use an existing IP address (if preconfigured) or access the network to configure a new one (the decision is left to implementation).

The following new error codes are also defined in the document and will be used in the Socket API in compliance with [RFC5014].

EAI_REQUIREDIPNOTSUPPORTED /* The network does not support the ability to request that specific IP address type */

EAI_REQUIREDIPFAILED /* The network could not assign that specific IP address type */

4. Usage example

The following example shows the code for creating a Stream socket (TCP) with a Session-Lasting source IP address:

```
#include <sys/socket.h>
#include <netinnet/in.h>

int              s ;                // Socket id
sockaddr_in6     serverAddress ;    // server info for connect()
uint32_t flags = IPV6_REQUIRE_SESSION_LASTING_IP ;
                                    // For requesting a Session-Lasting
                                    // source IP address

  // 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 (setsockopt(s,
                   IPPROTO_IPV6,
                   IPV6_ADDR_PREFERENCE,
                   (void *) flags,
                   sizeof(flags)) == 0){

          // setting session continuity to Session Lasting is successful
```

```
          // The application can connect to the server

          // Set the desired server's port# and IP address
        serverAddress.sin6_port = serverPort ;
        serverAddress.sin6_addr = serverIpAddress ;

          // Connect to the server
        if (connect(s, &serverAddress, sizeof(serverAddress))==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 the request of a Session-Lasting source address was successful
    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 the socket was successfully created but a Session-Lasting source
      // address was not provided
}  // 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 new flags will use the legacy API to the IP stack and will not enjoy On-Demand Mobility feature.

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

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 Mobility feature, the IP stack must respond in a legacy manner.

If the network infrastructure supports On-Demand Mobility feature, 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 Mobility feature. How the IP stack on the host and the network infrastructure behave in case of a compatibility issue is outside the scope of this API specification.

6. Summary of New Definitions

The following list summarizes the new constants definitions discussed in this memo:

<netdb.h>	IPV6_REQUIRE_FIXED_IP
<netdb.h>	IPV6_REQUIRE_SESSION_LASTING_IP
<netdb.h>	IPV6_REQUIRE_NON_PERSISTENT_IP
<netdb.h>	IPV6_REQUIRE_SRC_ON_NET
<netdb.h>	EAI_REQUIREDIPNOTSUPPORTED
<netdb.h>	EAI_REQUIREDIPFAILED
<netinet/in.h>	IPV6_REQUIRE_FIXED_IP
<netinet/in.h>	IPV6_REQUIRE_SESSION_LASTING_IP
<netinet/in.h>	IPV6_REQUIRE_NON_PERSISTENT_IP
<netinet/in.h>	IPV6_REQUIRE_SRC_ON_NET
<netinet/in.h>	EAI_REQUIREDIPNOTSUPPORTED
<netinet/in.h>	EAI_REQUIREDIPFAILED

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 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-05 (work in progress), October 2016.

[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

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 11, 2017

D. Moses
Intel
A. Yegin

February 7, 2017

DHCPv6 Extension for On Demand Mobility exposure
draft-moses-dmm-dhcp-ondemand-mobility-05

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 (or address), and networks to indicate the type of mobility service associated with the allocated IP prefix (or address) in return.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 11, 2017.

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 (or address) continuity after an event of the host moving to different locations with different points of attachments within the IP network topology. It further specifies means for applications to convey to the IP stack in the mobile host, their requirements regarding these services.

This document defines extensions to the DHCPv6 protocol ([RFC3315]) in the form of a new DHCP option that specifies the type of mobility services associated with an IPv6 prefix (or address). 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 (or address) in return.

This new option also extends the ability of mobile routers to specify desired mobility service in a request for IPv6 prefixes (as specified in [RFC3633]), and delegating routers to convey the type of mobility service that is committed with the allocated IPv6 prefixes in return.

It is important to note that although this document specifies extensions to both IPv6 address and IPv6 prefix assignments, host must follow the recommendations and guidelines listed in [RFC7934]

and attempt to obtain IPv6 prefixes (rather than addresses) from the network.

In a distributed mobility management environment, there are multiple Mobility Anchors (as specified in [I-D.ietf-dmm-distributed-mobility-anchoring]). In some use-cases, mobile hosts may wish to indicate to the network, their preference of the serving Mobility Anchor. This document specifies a new DHCPv6 option that is used by DHCPv6 clients to convey this preference.

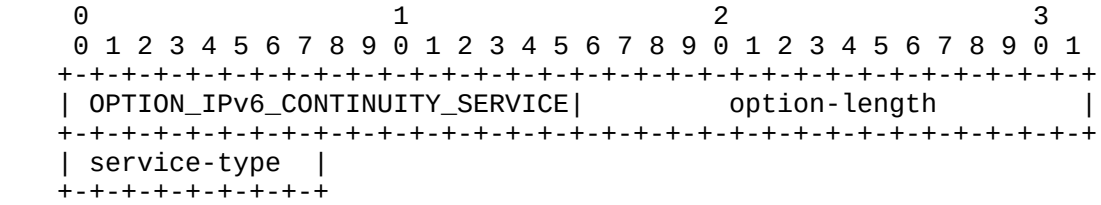
2. Notational Conventions

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

3. IPv6 Continuity Service Option

The IPv6 Continuity Service option is used to specify the type of continuity service associated with a source IPv6 address or IPv6 prefix. The IPv6 Continuity Service option must be encapsulated in the IAaddr-options field of the IA Address option when associated with an IPv6 address, and in the IAprefix-options field of the IA_PD prefix option when associated with an IPv6 prefix.

The format of the IPv6 Continuity Service option is:



option-code OPTION_IPv6_CONTINUITY_SERVICE (TBD)

option-len 1

service-type one of the following values:

Non-Persistent - a non-persistent IP address or prefix (1)

Session-Lasting - a session-lasting IP address or prefix (2)

Fixed - a fixed IP address or prefix (3)

Anytype - Anyone of the above (0)

The definition of these service types is available in [I-D.ietf-dmm-ondemand-mobility].

All other values (4-255) are reserved for future usage and should not be used. 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.

This option can appear in one of two contexts: (1) As part of a request to assign a source IPv6 address of the specified mobility service type, and (2) As part of a request to assign an IPv6 prefix of the specified mobility service type.

3.1. Source IPv6 Address Type Specification

In this context, the IPv6 Continuity Service option is encapsulated in the IAaddr-options field of the IA Address option.

When in a message 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 address requested by the client.

When in a message sent from a server to a client, the value of the IPv6 Continuity Service option indicates the type of IP continuity service committed by the network for the associated IPv6 address. The value 'AnyType' can 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 address type was requested and provided, any subsequent messages involving this address (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 received a request to assign an IPv6 address with a specified IPv6 Continuity service, but cannot fulfill the request, it must reply with the NoAddrsAvail status (refer to section 22.13 - Status Code Option in [RFC3315].

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

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

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

If a client receives an IA Address option from a server with the IPv6 Continuity Service option in the IAaddr-options field, without initially requesting a specific service using this option, it must discard the received IPv6 address.

If the mobile host has no preference regarding the type of continuity service it uses the 'AnyType' value as the specified type of continuity service. The Server will allocate an IPv6 address with some continuity service and must specify the type in IPv6 Continuity Service option encapsulated in the IAaddr-options field of the IA Address option. The method for selecting the type of continuity service is outside the scope of this specification.

3.2. IPv6 Prefix Type Specification

In this context, the IPv6 Continuity Service option is encapsulated in the IAprefix-options field of the IA_PD prefix option.

When in a message 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 in a message 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' can 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 was 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 received a request to assign an IPv6 prefix with a specified IPv6 Continuity service, but cannot fulfill the request, it must reply with the NoAddrsAvail status.

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

If a client does not receive the requested prefix, it must resend the request without the desired IPv6 Continuity Service option since it is not supported by the server. In that case, the requesting device (host or router) cannot assume any IP continuity service behaviour for that prefix.

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. Anchor Preference Option

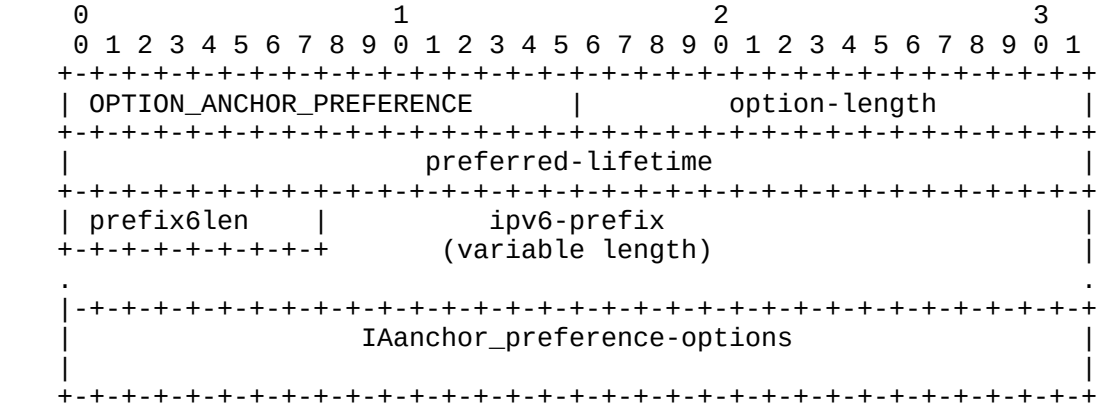
In a distributed mobility management environment that deploys multiple Mobility Anchors, each Mobility Anchor may have a set of IPv6 prefixes that is being used when assigning Session-lasting or Fixed source IPv6 prefixes (or addresses) to hosts.

The selection of the Mobility Anchor that will serve a mobile host is performed by the network at various events like, the event of initial attachment of a mobile host to a network.

The Anchor Preference option enables a host to express its desire to receive a specific source IPv6 prefix (or a source IP address with that specifc prefix). This is useful when the mobile host wishes to indicate to the network which Mobility Anchor should be used for anchoring its traffic and ensuring service continuity in the event of handoff between LANs with different IPv6 prefixes.

The network MAY respect this request but is not required to do so.

The format of the Anchor Preference option is:



option-code OPTION_ANCHOR_PREFERENCE (TBD)

option-len 5 + length of ipv6-prefix field + length of anchor_preference-options field

preferred-lifetime The preferred lifetime of the IPv6 address whose prefix is requested, expressed in units of seconds

prefix-length The length in bits of the ipv6-prefix. Typically allowed values are 0 to 128.

IPv6 prefix This is a variable length field that specifies the desired ipv6 prefix. The length is (prefix6len + 7) / 8. This field is padded with 0 bits up to the nearest octet boundary when prefix6len is not divisible by 8.

IAanchor_preference-option Options associated with this request

An IPv6 prefix (or address) is requested only when the mobile host wishes to be anchored by a specific mobility anchor. The client must also indicate the type of mobility service it requires using the IPv6 Continuity Service option encapsulated in the IAanchor_preference-options field of the IA_PD Prefix Option (or IA Address option - in the case of requesting an IP address).

When requesting a specific IPv6 prefix, only the ’Session-Lasting’ and ’Fixed’ types are legal. Non-Persistent prefixes do not require anchoring and there is no motivation to request a specific anchor.

If a server received a request to use a specific IPv6 prefix and an IPv6 Continuity Service type, but cannot assign an IPv6 prefix (or address) with that specified IPv6 Continuity Service it must reply with the NoAddrsAvail status.

A server that does not support this option will discard it.

A server is not required to respect the prefix request. It can assign a different prefix (or address) as long as it fulfills the IP Continuity Service request.

If a client does not receive any address, it must assume that the the option is not supported by the server and use the IA Address option in subsequent requests.

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>.

7.2. Informative References

[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-03 (work in progress), December 2016.

[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-10 (work in progress), January 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

Alper Yegin
Istanbul
Turkey

Email: alper.yegin@yegin.org

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 28, 2017

B. Pularikkal
Cisco Systems
Q. Fu
H. Deng
China Mobile
G. Sundaram
S. Gundavelli
Cisco Systems

February 24, 2017

Virtual CPE Deployment Considerations
draft-pularikkal-virtual-cpe-02

Abstract

Broadband Service Provider Industry has been gearing towards the adoption of Virtual CPE (vCPE) solutions. The concept of vCPE is build around the idea that the physical CPE device at the customer premises can be simplified by moving some of the key feature functionalities from the physical CPE device to the Service Provider Network. This document starts discussing the drivers behind vCPE adoption followed by Solution level requirements. Two key Architecture models for vCPE, which can address the service provider and subscriber requirements, are covered in this reference document. Document also touches up on some of the key deployment considerations, which can influence the adoption of the vCPE architecture models.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 28, 2017.

Table of Contents

1. Introduction

Broadband Service Providers are constantly looking for opportunities to generate additional revenue streams from their existing broadband infrastructure. In order to achieve this, new value added services need to be created for the end customers. Customer retention is another key focus area for broadband subscribers, where they have been facing competition from Internet content providers on home multi-media services such as broadcast video, video on demand and voice. There is a need to improve the overall end user experience on an ongoing basis to reduce the subscriber churn. In order to achieve these business goals, Broadband Service Providers are starting to

consider the deployment of Virtual CPE based Architectures. There are several factors, which are driving the adoption of vCPE-based solutions. Also the recent technological advancements in cloud computing and software defined networking are expected to further accelerate the adoption vCPE based architectures.

The key aspect of the vCPE solutions is the simplification of the physical CPE device. Such a simplification allows minimizing the feature dependency on CPE vendors for the roll out of new service offerings. Also it reduces the complexities around service provisioning, Service Upgrade Troubleshooting etc. There are multiple architecture options being considered by the industry for vCPE solutions.

Objective of this draft is to serve as a reference material for Broadband Service Operators who are interested in migrating to vCPE based architectures. The document starts with going over some of the key drivers for vCPE solution adoption. Also it covers typical solution level requirements, which needs to be considered while selecting the right architecture models. Document also touches up on some of the key deployment considerations, which can influence the adoption of the vCPE architecture models.

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].

3. Solution Requirements for vCPE

This section provides a high level summary of solution requirements, which needs to be addressed by Virtual Connected Home Architecture Models. The solution requirements can be broadly classified under the following categories:

(1) Subscriber side requirements: Subscriber in the context of this documentation refers to a homeowner with Broadband connection. These requirements primarily map to the end user experience for a home subscriber in terms of connectivity, quality of experience and value added services.

(2) Broadband Operator side requirements: Operator is the broadband service provider such as Cable MSOs, DSL providers etc. These requirements primarily maps to the business aspects which needs to be covered in the solution in terms of CAPEX, OPEX reduction, service velocity, new revenue generation opportunities etc.

High level requirements under the above two categories are summarised in the table below:

Subscriber Side Requirements	Operator Side requirements
1) Private Home Network	1) Service Velocity
2) Zero Touch Provisioning	2) Simplified CPE
3) Local Bridging	3) Per UE Visibility
4) Local Routing	4) Community Wi-Fi
5) NAT, FW, IDS, Parental Control	5) IP Address Persistence
6) Home Network Analytics	6) UE Attachment/ Detachment detction
7) Self Service Portal	7)Usage based billing
8) Dynamic IP address Assignment	8)Quality of Service
9) Home Network Remote Access	9) NAT Traversal

Figure 1: VCPE Requirements

4. Architecture Models for vCPE

In this document three different architecture models are covered for the Virtual CPE based solutions. This section starts with a definition of what represents a virtual CPE and then gets into the details of the Architecture options, which are available for the implementation of the same.

4.1. Virtual CPE Definition

A virtual CPE (vCPE) is a logical representation of classical CPE functions performed by a physical CPE device. In other words, business logic and feature functionalities which are traditionally embedded in a CPE device is separated from the hardware device and runs in the Service Providers cloud. Concepts of vCPE has basis on the Network Function Virtualization. The business logic and feature functionalities of a CPE device are virtualized and runs as NFV in the cloud. Each simplified physical CPE would have a corresponding virtual CPE function running in the cloud. There are several ways to realize this vCPE instance in the cloud. One approach is to have separate vCPE instance running as a Linux container or micro-VM corresponding to each physical CPE instance. The vCPE may also be implemented as a representational state on aggregation platforms such as broadband network gateways (BNGs). A third approach may rely on a

combination of the BNG representational state and Service function chaining to represent the vCPE instance in the cloud. These Architecture models are covered in the subsequent sub-sections.

4.2. Virtual CPE Architecture Model-01

This model is build around the concept of separate virtualized instance per physical CPE device. In this model Virtual CPE instance handles the control plane as well as the data plane. Each micro VM represents an NFV element of CPE with integrated control and data plane. All feature functionalities get implemented on the NFV element itself. This model does not leverage the dynamic service function chaining capabilities.

A high level Architecture view of this model is provided in figure-below:

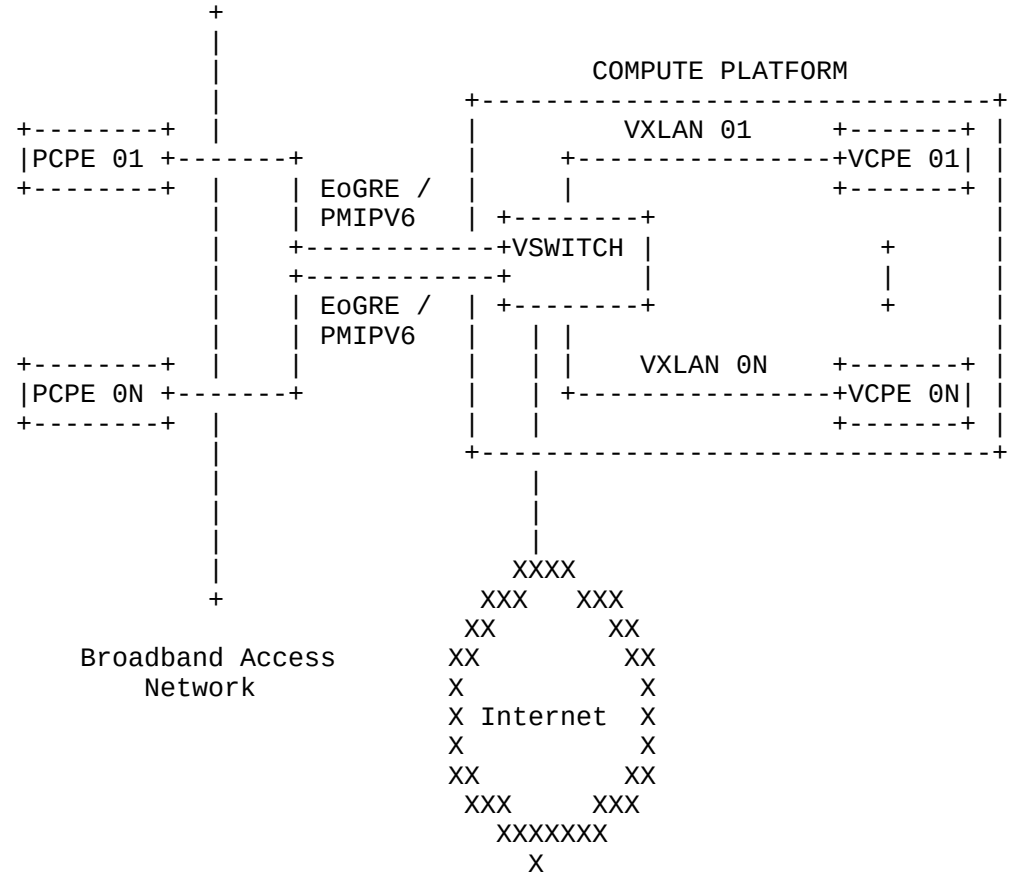


Figure 2: VCPE deployment model-01

P-CPE device performs just the bridging function where the layer-2 traffic between directly connected devices will be simply bridged by the P-CPE. Any layer-3 traffic will be transparently forwarded to the cloud over the tunnel.

In this model all the key cloud based components run on virtualized platforms. These virtual components are deployed on standalone virtual platforms rather than on large scale virtual DC of Service providers. Therefore, the tunnel will be terminated on the vSwitch rather than a tunnel termination GW located at the boarder of the DC.

An implementation could leverage either a vendor specific vSwitch or an Open vSwitch.

Tunnel end points are uniquely identified with the IP address of the P-CPE. The vSwitch maps the de-encapsulated traffic from the tunnels to unique VXLANs and will forward to the corresponding Micro VM instances. Micro VM instances will be responsible for supporting the key functions traditionally performed on physical CPE devices. After the feature processing, V-CPE instance will send the traffic back to the v-Switch over VXLAN tunnels and vSwitch will forward it to external network.

4.3. Virtual CPE Architecture Model-02

In this model vCPE in the cloud is corresponding to each physical CPE is realized by a representational state on a tunnel aggregation platform such as BNG. A provisioned physical CPE in run state is expected to have at least one tunnel established between the physical CPE and the BNG. As long as the PCPE is in run state there will be a CPE session on the BNG which represents the CPE itself. Some of the key CPE features will be running on the BNG while supplementary features and services can be deployed using dynamic service function chaining functions.

A high level Architecture view of this mode2 is provided in figure-below:

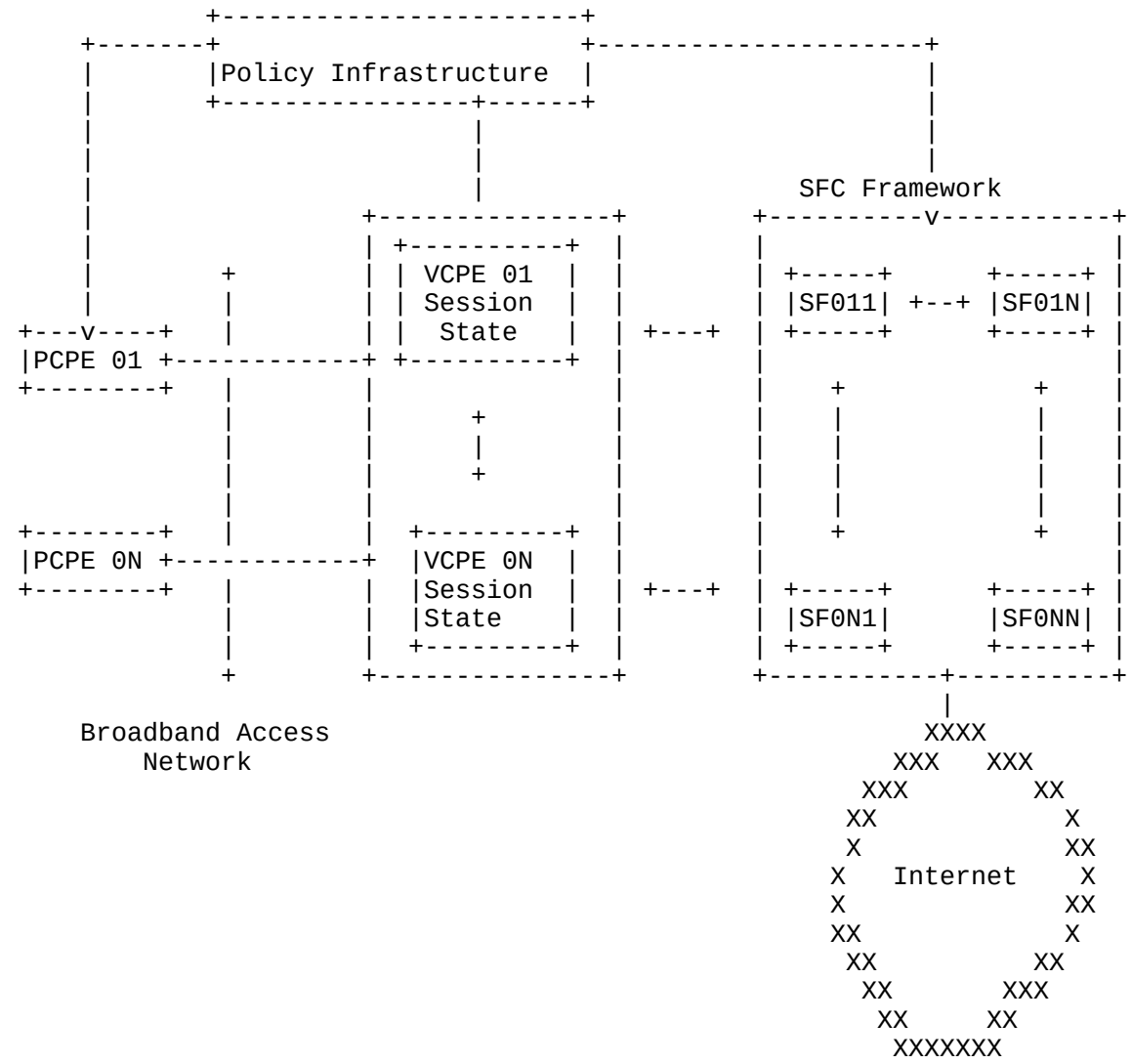


Figure 3: VCPE deployment model-02

In this model as well, P-CPE device performs just the bridging function where the layer-2 traffic between directly connected devices will be simply bridged by the P-CPE. Any layer-3 traffic will be transparently forwarded to the BNG over the tunnel.

There is no need for pre-configuration of the tunnels on BNG. When a P-CPE device become active and gets provisioned it will try to

establish an EoGRE tunnel session with V-CPE. Up on detecting a new P-CPE end point, the BNG would invoke an authorization process for the tunnel end point. It is up to the implementation to decide whether an out of band authentication mechanism is required before establishing v-CPE state on the BNG. If the access network is untrusted, the service provider may decide to overlay the EoGRE tunnel with IPSec encapsulation.

BNG will need to uniquely tag the subscriber flows before forwarding to the SFC framework. This can be accomplished by using some scalable tagging mechanisms such as VXLAN.

4.4. Virtual CPE Architecture Model-03

This is similar to model-02 but leverages split architecture for control plane and data plane for the BNG. This model introduces the concept of a BNG controller, which essentially carries out the control plane functions. Data plane component of the BNG can be a purpose built hardware optimized for scaleable tunnel termination, data encryption and data forwarding. Control plane intelligence of each vCPE resides as a session state on the BNG controller and the data plane intelligence including tunnel termination of each vCPE resides on the BNG-DP system. BNG-CP will leverage the FPC (Forwarding Policy Configuration) interface which is being defined in the DMM working group to instruct the BNG-DP system to establish V-CPE DP states with relevant configuration.Role of FPC interface in this solution is described in the sub-section below. In this model, all basic and supplementary subscriber features will be implemented using a dynamic service function-chaining framework.

A high level Architecture view of this mode3 is provided in figure-below:

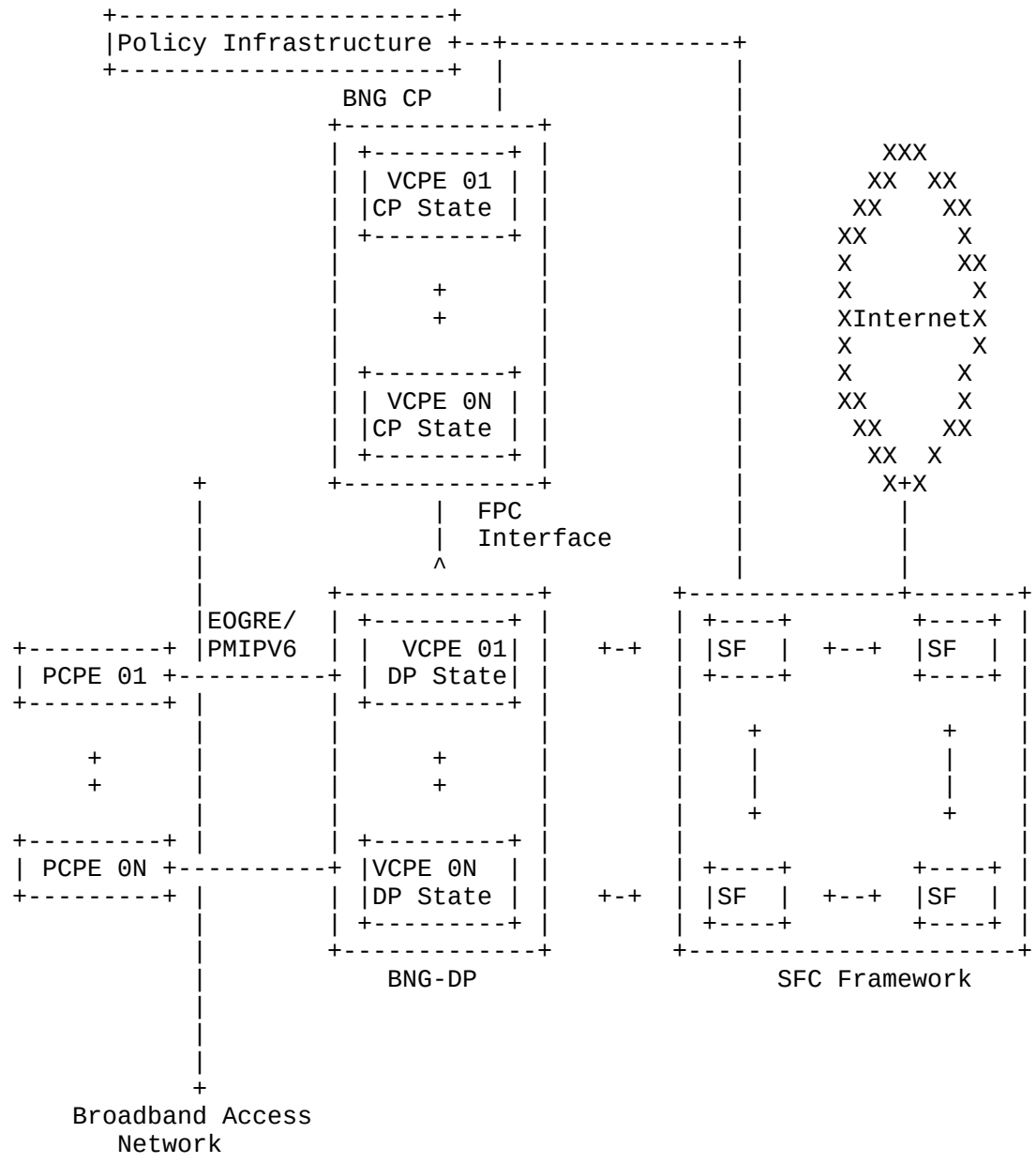


Figure 4: VCPE deployment model-03

4.4.1. Forwarding Policy Configuration (FPC) Interface

FPC Protocol interface defined in the DMM working group enables DMM use cases with Control Plane and data plane separation. In vCPE solution model-03, FPC protocol is applied to the interface between BNG-CP and BNG-DP. FPC interface consists of a client function which resides on the Control Plane System (BNG-CP in this case) and an agent function which resides on the Data Plane System (BNG-DP in this case). FPC defines a standard set of protocol semantics to exchange configuration information from the client to the agent. Agent processes the protocol semantics and translates them into configuration commands as per BNG-DP system technology. FPC client function residing on BNG-CP device will leverage FPC Protocol semantics to provision activate or deactivate the V-CPE DP states on BNG-DP with desired features.

Some of the FPC attributes needed for vCPE implementation are listed in the tables below:

Attribute	Description	Availability
PROP_TUN	Tunnel Encapsulation Type	Defined in FPC
PROP_GW	Default Gateway IP Address for client traffic	Defined in FPC
PROP_NSH	Add NSH Header	Defined in FPC
PROP_PUNT	Default next hop for unknown traffic class	Not in FPC Draft
PROP_L2PASS	Layer 2 passthrough for the matching traffic	Not in FPC Draft
PROP_QOS_GBR	Guaranteed bit rate for a port or port group	Defined in FPC
PROP_QOS_MBR	Maximum bit rate for a port or port group	Defined in FPC
PROP_DROP	Drop the IP packet	Defined in FPC
PROP_DROP_L2	Drop the layer 2 packet	Not in FPC Draft

Figure 5: FPC Attributes needed for vCPE Model-03

Attributed listed in the table above just represents a sample list. The complete list of attributes will be defined in a companion draft.

5. Deployment Considerations for vCPE

This section at a high level touches up on some of the key deployment charecteristics which needs to be considered while selecting the right vCPE architecure

5.1. Multi-tenancy

vCPE represents the abstraction of key functions and features typically performed by classical device into service provider cloud. In order for such a solution to be operationally feasible and profitable, it is important for vCPE architecture to support multi-tenancy. This multi tenancy support needs to scale of the order of

hundreds of thousands. From the context of vCPE deployments, the multi-tenancy refers to the logical separation of vCPE instances, which are housed in a common backend infrastructure. This backend infrastructure could consist of virtual elements on a compute platform or physical networking components. It could very well be a combination of virtual and physical components in the service provider cloud. Few of the key areas where multi-tenancy model will have an implication on the operational efficiency of the solution are listed below:

Overlapping IP addressing: Typically home networks are configured with RFC 1918 private address space 192.168.0.0/24. A vCPE solution, which deals with IP address management of the private home network, must support address overlap for these private home subnets.

Tunnel scale: Tunnel termination points in the service provider must support tunnel scale of the order of hundreds of thousands. A vCPE implementation must implement some form of unique tunnel id per physical CPE to support saleable multi-tenancy for tunnel termination.

Overlapping SSID naming: vCPE framework must be flexible enough to allow home subscribers to configure private SSID names of their choice. Possibility of overlapping SSID names cannot be ruled out as subscribers randomly decide up on their private SSID names. Multi-tenancy solution for a vCPE framework must take into consideration this.

5.2. Tunneling

In a vCPE solution, the end subscriber data must be tunneled from the physical CPE towards the vCPE instance in the cloud. Typical home broadband deployments may have community Wi-Fi SSID enabled in addition to subscribers private home SSID. For such cases, the tunnel must be capable of carrying both private and community Wi-Fi SSID traffic in a secured manner. Today there are various tunneling methods being used for community Wi-Fi deployments. Two of the most common tunneling methods in use are EoGRE and PMIPv6. EoGRE is a layer-2 tunneling technology and it does not have a control plane of its own. PMIPv6 is a layer-3 tunneling technology with a well-defined control plane for tunnel management and session management. Either of these tunneling options can be leveraged to carry the private SSID traffic from the home towards the cloud-based vCPE. And both are capable of carrying community Wi-Fi and private home SSID traffic. The choice of the tunneling technology may be influenced by various factors such as simplicity, need for IP address persistence with client roaming, layer-2 forwarding in the data plane to the cloud as opposed to layer-3 forwarding etc.

5.3. Security

The classical home broadband deployments based up on intelligent physical CPE devices typically provide data privacy and security for the end subscriber content as it gets carried over the access network. A security framework for a vCPE network has to account for the following key aspects:

Subscriber Authentication

Protection against spoofing attacks

Data privacy

Prevention of eaves dropping between subscribers

Security considerations go hand in hand with multi-tenancy requirements as data and meta data from multiple subscribers will be handled by the backend systems.

5.4. Dynamic Service Chaining

One of the key motivation behind a cloud based connected home solution is to find additional revenue generation opportunities through rapid deployment of new services. The implementation of these new services requires a combination of system and network level functions to be applied to the end user traffic flows. Some of these functions may be enabled, by leveraging system level features on the CPE Device Anchor. But in many cases, it makes more sense to offload the feature processing to network function elements, which are external to the CPE Device Anchor (CDA).

Service function chaining (SFC) refers to a collection of network elements connected in a serialized fashion through which a traffic flow will be diverted prior to forwarding to the intended destination. Traditionally these service chains are hard connected there by causing challenges around flexibility and scale.

With dynamic service function chaining approach, the network elements, which perform various service functions, are arranged in grid model. Logical connectivity is established on a per traffic flow basis between the network elements to establish SFC pipeline for a qualified traffic flow. Dynamic SFC addresses the scale and flexibility limitations of the traditional chaining model. A vCPE solution must support the deployment of dynamic service function chaining.

5.5. NAT Traversal

Some vCPE deployments may leverage third party access networks and offer the solution as an overlay. In such cases, there may be requirement to bring up P-CPE behind a NAT router. The vCPE service provider may not have direct control over the NAT router, which is managed by the access network provider. In such cases, a tunnel transport mode, which can traverse NAT, needs to be selected.

EoGRE tunnels do not support NAT traversal, since there is no UDP layer in the encapsulation header. PMIPv6 can support NAT traversal if the right data encapsulation option is selected. If a layer tunneling technology is desired for the implementation where NAT traversal is a requirement, then tunnel transport mechanisms such as L2TPV3 may be explored.

6. Conclusion

In this document, the concept of VCPE is illustrated in detail. The basic concept of VCPE is to shift the complicated functions from the pCPE at the customer side to the VCPE at the service provider side. The motivation of such shifting can be concluded as providing quick launched customer defined services, reducing the Capex and Opex of the pCPE, and simlify the maintainance of both pCPE and VCPE. A use cases of community Wi-Fi is proposed for VCPE, which is a typical scenario for DMM. Three models are then discussed for the field deployment of VCPE. And CP/DP interface is suggested to be utilized in the deployment models.

7. Informative 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>.

Authors' Addresses

Byju Pularikkal
Cisco Systems
170 West Tasman Drive
San Jose
United States

Email: byjupg@cisco.com

Qiao Fu
China Mobile
Xuanwumenxi Ave. No.32
Beijing
China

Email: fuqiao1@outlook.com

Hui Deng
China Mobile
Xuanwumenxi Ave. No.32
Beijing
China

Email: denghui@chinamobile.com

Ganesh Sundaram
Cisco Systems
170 West Tasman Drive
San Jose
United States

Email: gsundara@cisco.com

Sri Gundavelli
Cisco Systems
170 West Tasman Drive
San Jose
United States

Email: sgundave@cisco.com

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

S. Jeon
Sungkyunkwan University

March 13, 2017

Stateless mobility functions
draft-sijeon-dmm-stateless-mobility-function-00.txt

Abstract

This draft presents two use cases to start a talk of stateless mobility function architecture in IETF DMM WG.

Table of Contents

1. Introduction

A mobility function could be categorized stateful function and stateless function. The stateful function maintains the state information associated with a terminal or network situation while the stateless function does not keep the state information but only focusing on processing received signaling messages or data packets, as a worker.

Combing the state with the worker in the mobility function has basically been considered in many network architectures for easier implementation and negligible latency for accessing the state database without external signaling messages. However, it is nowadays challenging as it tackles the flexibility for network scaling and other enhanced operation support.

Separating the control-plane and user-plane makes a progress for the flexible network control and provisioning. That is, a routing controller with holistic view can take a decision of forwarding behavior in the network entities. Stateless user-plane architecture is proposed in [I-D.matsushima-stateless-uplane-vepc], suggesting a mobile architecture that the user-plane network is governed by virtualized EPC decorated with mobility control-plane functions only. The state information the vEPC is holding is transferred by BGP routing to the user-plane architecture. Therefore, the user-plane architecture is totally abstracted and becomes state free, supporting high flexibility in network scaling.

Mobility architecture is composed of many control-plane functions. Scale-in/scale-out of those functions is of importance for elastic function resource handling and management (e.g., load balancing). User-plane functions does not generically hold the state information much, so it is relatively easier to do scaling but it is challenging to scale with mobility control-plane functions. Separating the state information from the control-plane functions could facilitate flexible function resource provisioning and expect further enhanced scenarios such as VNF mobility and migration procedure easier.

The main objective of this draft is to bring a broad discussion in IETF DMM WG, identifying what needs and requirements for the state separation from a mobility function are existing and what use cases could be meaningful, finally bearing a work item. In this draft, we start with two cases; i) mobility control-plane function integrated with the state information and ii) mobility control-plane function separated from the state information. From the two cases, we check the mentioned aspects.

2. Mobility function architecture

2.1. Integrated state with mobility function

Fig. 1 shows the state information is integrated in the mobility control-plane function. Suppose that the mobility control-plane function is composed of state and worker where the work is dedicated to the processing of signaling messages or data packets without concerning or maintaining the state. When the function is instantiated, the state is also initialized within the function. On the other hand, the function needs to be shut down due to some maintenance purpose and the same kinds of a new function is supposed to be initiated, the managed mobility state associated with mobile terminals should be moved to the newly initiated mobility control-plane function through a migration procedure in this case.

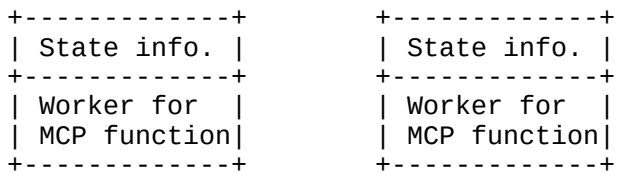


Fig. 1. The mobility function integrated with the state information

2.2. Separated state from mobility function

Fig. 2 shows the state information is separated from the mobility control-plane function, thus the worker for mobility control-plane function remains in the function. The relationship between the state database and worker can be defined by internal interface or external interface. Following the same scenario described in 2.1, for a need of replacing with a new worker or for scaling out/in, maintaining the state is not constrained, so facilitating various flexibility scenarios.

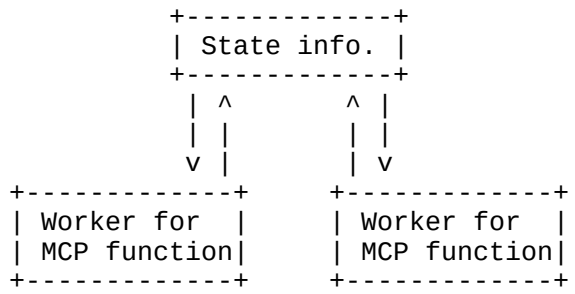


Fig. 2. The mobility function separated from the state information

3. IANA Considerations

This document currently makes no request of IANA.

4. Security Considerations

5. Informative References

[ETSI.NFV-VNFA]
"Network Functions Virtualisation (NFV); Virtual Network Functions Architecture, ETSI GS NFV-SWA 001, v1.1.1", Proceedings of GlobeCom Workshop on Seamless Wireless Mobility, December 2014.

[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.

Author’s Address

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

Email: seiljeon@skku.edu

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 24, 2017

Z. Yan
CNNIC
J. Lee
Sangmyung University
December 21, 2016

Mobility Ability Negotiation
draft-yan-dmm-man-00

Abstract

Based on IPv6, multiple mobility management protocols have been developed and generally they can be classified into two types: network-based and host-based. Different protocols have different functional requirements on the network element or the terminal and then a scheme should be used in order to support the negotiation and selection of adopted mobility management protocol when a terminal accesses to a new network. In this draft, this issue is considered and analyzed.

Table of Contents

1. Introduction

As the Mobile IPv6 (MIPv6) [RFC6275] protocol standardization suite, there are multiple extension protocols have been standardized. These protocols can be classified into two types: protocols for the function extension and protocols for the performance enhancement. The protocols for the function extension is 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 mobile nodes of multi-interface and Network Mobility (NEMO) [RFC3963] for mobility of sub-network. The other type is proposed to enhance the performance of the mobility management, such as Fast Mobile IPv6 (FMIP6) [RFC5268] for fast handover, Hierarchical Mobile IPv6 (HMIPv6) [RFC5380] for hierarchical mobility optimization. MIPv6 and these extensions are called host-based mobility management protocols because the location update is initiated by the terminal and the tunnel is terminated at the terminal.

In order to reduce the protocol cost and enhance the handover performance further, the network-based mobility management schemes were proposed and Proxy Mobile IPv6 (PMIPv6) [RFC5213] was standardized as a basis. Based on PMIPv6, a series of its extensions were proposed, such as Dual-stack Proxy Mobile IPv6 (DS-PMIPv6) [RFC5844], Distributed Mobility Management Proxy Mobile IPv6 (DMM-PMIPv6) [RFC7333] and so on. Be different from the host-based schemes, the location update in PMIPv6 is triggered by the network entity and the tunnel is established between network entities. And the terminal needs to do nothing about the signaling exchange during the movement, particularly, the mobility is transparent to the IP layer of the terminal.

In reality, these protocols will be co-existing and multiple protocol daemons will be configured on the network entities or terminal. That means a scheme is needed to support the negotiation and selection of mobility management protocol when the terminal accesses into a new network initially or handover happens.

This document tries to analyze this problem with the possible scenarios should be supported by the further solution.

2. Motivation

As illustrated above, these protocols may co-exist in reality and simultaneously used in an access network or even the same entity. Due to their different requirements on the network entity or terminal, a scheme is needed to support the negotiation and selection of adopted mobility management protocol when the terminal accesses to a new network.

Generally, two problems should be solved:

- Which entity (network or terminal) will initiate the protocol negotiation and selection?
- What principles should be followed for the protocol negotiation and selection?

This scheme is needed because different entity and terminal will have different abilities and preferences (may be decided by the ability and mobility pattern of the mobile node). This scheme can guarantee that the optimum and most suitable protocol will be used.

3. Scenarios

In order to illustrate the necessity of the mobility ability negotiation and protocol selection, the following scenarios are taken as typical examples:

1) Network supports MIPv6, host supports only PMIPv6

In this case, the network supports only host-based scheme, while the host does not have any mobility management function. Then only the PMIPv6 can be used to support the mobility management of the host, but the network does not deploy the PMIPv6 protocols and this will cause the mobility failure because no available protocol can be used.

2) Network supports both MIPv6 and PMIPv6, host supports only PMIPv6

In this case, the network deploys both host-based and network-based schemes, while the host supports only PMIPv6. When the host accesses

to the network, PMIPv6 will be used. Actually, PMIPv6 should be adopted as a default mobility management scheme considering its optimized performance. Once the PMIPv6 can be supported by the network, it will be adopted as the default scheme.

3) Network supports both MIPv6 and PMIPv6, host supports MIPv6

In this case, the network deploys both host-based and network-based schemes, and the host also supports MIPv6. Because PMIPv6 has no requirement on the host, both PMIPv6 and MIPv6 can be used in this case. Then the host can use PMIPv6 as default, and use MIPv6 for the global mobility.

4) Network and host support all the extention protocols

In this case, the network has deployed multiple extensions to support more complex requirements, so does the host. And then the host and network can negotiate a protocol for the optimized performance or special requirement, for example, FMIPv6 may be selected in order to support fast handover, HMIPv6 may be selected in order to reduce the signaling cost, NEMO may be selected in order to support the subnet mobility.

4. Possible solutions

Two different schemes may be used for the protocol negotiation and selection: MN-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 and general procedure may be followed:

- During initiation, PMIPv6 may be used as a default mobility management protocol once the network supports it.
- If the host prefers host-based scheme, a negotiation is executed to handover from PMIPv6 to MIPv6 style.
- After initial attachment, a profile will be generated in the management store to record the selected protocol of this host.
- When the handover happens, the network will check the selected protocol during the authentication process. But the network also needs to notify the host if the selected protocol cannot be supported herein.

In order to fulfill the above principles, some extensions should be supported, for example:

1) Extended negotiation messages

In order to negotiate the mobility management protocol between host and network, a new signaling message or an extended message (e.g., ICMPv6, Diameter, and RADIUS) should be used. Except 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 terminal accesses to the network, an 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.

5. Security Considerations

Generally, this function will not incur additional security issues. The detailed influence should be analyzed in the future.

6. IANA Considerations

A new ICMP option or authentication option or other signaling message may be used with a new code number.

7. References

[RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, DOI 10.17487/RFC3963, January 2005, <http://www.rfc-editor.org/info/rfc3963>.

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

[RFC5268] Koodli, R., Ed., "Mobile IPv6 Fast Handovers", RFC 5268, DOI 10.17487/RFC5268, June 2008, <http://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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc5844>.

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

[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

Zhiwei Yan
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing 100190
China

Email: yan@cnnic.cn

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

Email: jonghyouk@smu.ac.kr